UNIVERSITÉ DE SHERBROOKE

Faculté des sciences appliquées

Département de génie électrique

CONVERSION À HAUTE RÉSOLUTION D'UN SIGNAL ENTRELACÉ

Mémoire de maîtrise ès sciences appliquées

Spécialité : génie électrique

Ba-Thao Nguyen

Sherbrooke (Québec), CANADA

Octobre 1996

DÉDICACE

À mes chers parents

À mes frères et mes soeurs

SOMMAIRE

Le but de cette recherche appliquée est de trouver une méthode pratique pour augmenter la résolution des images de télévision entrelacées sans recourir à des changements radicaux dans la transmission des signaux. Il s'agit donc ici de faire passer le signal entrelacé à travers une série de filtres non linéaires et multidimensionnels. Ce processus, qualifié de «**dé-entrelacement d'une séquence d'images entrelacées**», produit des images progressives que l'on peut ensuite surconvertir en des formats de Télévision à Haute Définition (TVHD). Aussi faut-il concevoir un nouveau système de conversion efficace et fonctionnant parfaitement en temps réel.

Après un rappel des techniques de conversion existant actuellement, on aborde en profondeur les principes et les caractéristiques du système proposé.

Le système présenté est original en ce qu'il applique en plus de trois méthodes de conversion de signal, à savoir la méthode de détection de contour, celle de détection de mouvement et la méthode du filtrage adaptatif, les techniques des moyennes et surtout celles de renforcement de décisions séquentielles.

REMERCIEMENTS

Permettez-moi d'exprimer ici ma profonde gratitude envers les personnes qui m'ont appuyé dans la réalisation de ce mémoire. Mes remerciements particuliers vont à:

Monsieur Chon Tam Le Dinh, mon directeur de recherche, pour ses conseils judicieux, sa disponibilité et son soutien financier.

Messieurs Christian Tremblay et Marcel Auclair de la Compagnie Miranda inc. pour avoir initié et appuyé financièrement ce projet.

Messieurs André Vincent et André Mainguy du Centre de Recherche sur les Télécommunications du Canada (CRTC) pour avoir mis à notre disposition le matériel requis ainsi que pour leurs précieux commentaires.

Mes collègues de travail pour leur complicité et leurs échanges fructueux.

Et pour terminer, un remerciement profond à mes parents, frères, soeurs et amis qui ont su m'encourager à persévérer dans cette entreprise.

TABLE DES MATIÈRES

LISTE DES FIGURES ET DES TABLEAUX

1

INTRODUCTION

Le monde assiste actuellement à une période de transition et de changements dans le domaine des télécommunications visuelles. En effet, la télévision évoluera progressivement vers la haute définition (TVHD) au cours des prochaines années.

Comme on ne peut pas changer radicalement les postes d'émetteur et de récepteur pour recevoir la TVHD, la période de coexistence simultanée de deux systèmes TVHD et NTSC sera échelonnée sur au moins dix ans.

Entre temps, les chercheurs se concentrent sur des techniques de conversion et de traitement d'images dans le but d'obtenir des images quasi-TVHD à partir des signaux NTSC. Ces techniques resteront toujours utiles dans l'avenir lorsque l'on voudra convertir de vieux films ou de vieux vidéos en images progressives.

Les systèmes actuels présentent des faiblesses que l'on peut regrouper en trois catégories:

i) les limitations du monochrome, structure des lignes visibles (525 lignes, entrelacement 1:2);

ii) la limitation du codage NTSC, spécialement celle reliée à l'interférence entre la luminance et les chrominances modulées;

iii) les problèmes de GAMMA qui usurpent le principe de luminance constante [1], [2], [3].

Le présent travail s'oriente uniquement sur l'amélioration de la résolution d'image, c'est-à-dire que le traitement est fait à partir des signaux composants (la séparation complète des signaux de luminance et de chrominance est requise présumée). Dans ce cas, l'interpolation est appliquée de

façon indépendante pour les composantes de la luminance Y et des chrominances U, V. Les signaux composés comme NTSC, PAL et SECAM nécessitent alors un décodeur approprié.

Étant donné que l'oeil est plus sensible aux signaux de luminance qu'à ceux des chrominances, on travaille surtout avec la composante Y, si le système fonctionne bien, on peut le généraliser pour les composantes U, V.

Le doubleur de lignes proposé dans la présente étude est principalement basé sur un détecteur de contour, un détecteur de mouvement et des filtres adaptatifs à trois dimensions (3D). En particulier, la détection de contour est basée sur les cinq directions les plus fréquentes dans une image. Elle intervient dans le choix des filtres appropriés dans le domaine vertical-temporel.

Ce mémoire comporte cinq parties. Après une brève introduction du sujet, le chapitre 2 traite des différentes techniques de conversion existantes et de l'approche de Zenith pour la TVHD. Le chapitre 3 présente le nouveau système proposé ainsi que le principe et le fonctionnement de chaque sous-système. Les côtés performance et pratique de ce système y sont mis en évidence. Le chapitre 4 est consacré à la conversion en formats TVHD de l'image progressive obtenue avec le système proposé dans le chapitre 3. En plus, on y décrit la technique de filtrage non séparable (V-T).

Enfin, le dernier chapitre expose les résultats de la simulation du système proposé, ainsi que ceux obtenus par des approches conventionnelles. Le lecteur ou la lectrice y aura l'occasion d'examiner les images prises directement à l'écran d'une station Sun.

2

LES TECHNIQUES DE CONVERSION EXISTANTES

Le standard nord américain de TVHD a été défini récemment par le consortium «Grand Alliance». Selon ce dernier, l'image sera d'abord structurée en 787,5 lignes progressives et à long terme en 1050 lignes progressives. Il est donc important de convertir des images vidéo existantes qui approchent la norme et la qualité des images TVHD. Dans la conversion de signal vidéo conventionnel à signal vidéo TVHD, plusieurs chercheurs ont proposé d'utiliser un doubleur de lignes, suivi d'un interpolateur vertical d'un rapport approprié. La performance du doubleur de lignes est primordiale; ainsi cette recherche est-elle orientée vers les techniques de conversion du signal entrelacé en signal progressif. Ces techniques impliquent essentiellement des opérations d'interpolation.

L'interpolation est en général un processus de reconstitution qui fait naître une variable de sortie Y inexistante à partir d'un ensemble de variables X_i d'entrée. En traitement d'image, elle est équivalente à l'opération qui crée des lignes manquantes dans le champ pair (ou impair) du signal vidéo. Les techniques d'interpolation retenues sont de deux catégories: l'interpolation par filtrage linéaire et l'interpolation par filtrage non linéaire.

2.1 Interpolation par filtrage linéaire

L'interpolation par le filtrage linéaire est bien connue dans la littérature du traitement du signal. Elle se compose de deux opérations successives. Il s'agit de l'échantillonnage surélevé (upsampling) qui introduit des zéros entre deux échantillons existants et du filtrage passe-bas linéaire pour supprimer les composantes à haute fréquence dues à l'échantillonnage surélevé. D'une part, dans le traitement vidéo en temps réel, les filtres conçus ne doivent pas être complexes mais

efficaces. D'autre part, les filtres spatiaux doivent être linéaires de phase. Considérons maintenant quelques filtres d'interpolation de lignes.

2.1.1 Interpolation par un filtre vertical

Les filtres verticaux sont très faciles à concevoir et n'ont besoin que de peu de mémoire du système. Par contre, l'interpolation verticale seule cause souvent l'effet d'escalier lorsque le signal original subit déjà un sous échantillonnage sévère.

2.1.1.1 Interpolation par un filtre vertical d'ordre zéro [15]

Comme on sait que l'image entrelacée est composée de deux champs ou demi-trames, la décomposition doit se faire avant l'interpolation. La figure 2.1 donne un exemple de ce processus.

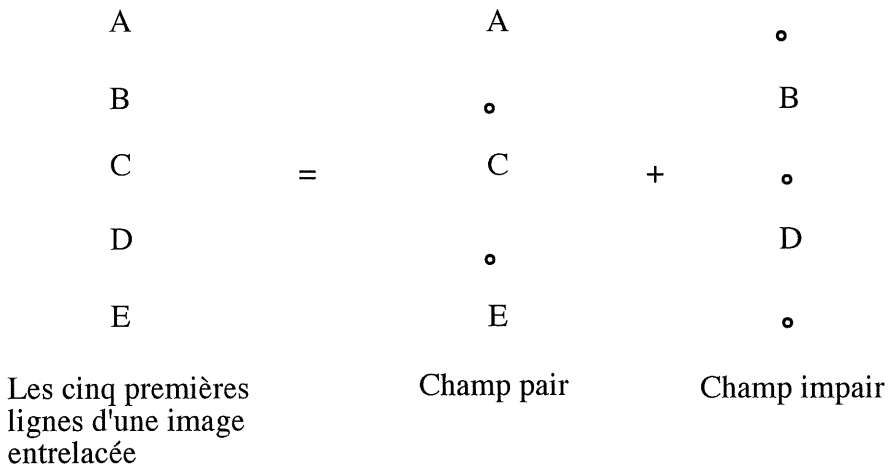
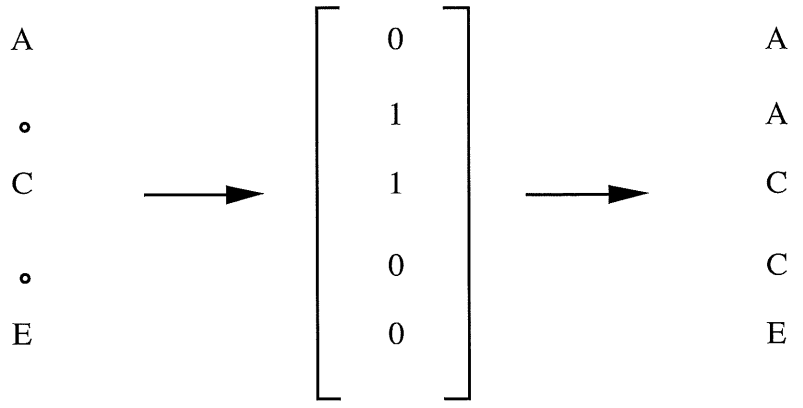


Figure 2.1 La décomposition d'une image entrelacée

En séparant deux champs d'une image entrelacée, on se trouve avec deux images, dont chacune contient des lignes inexistantes. La plus simple approche consiste à répéter la ligne existante, comme le montre la figure 2.2.



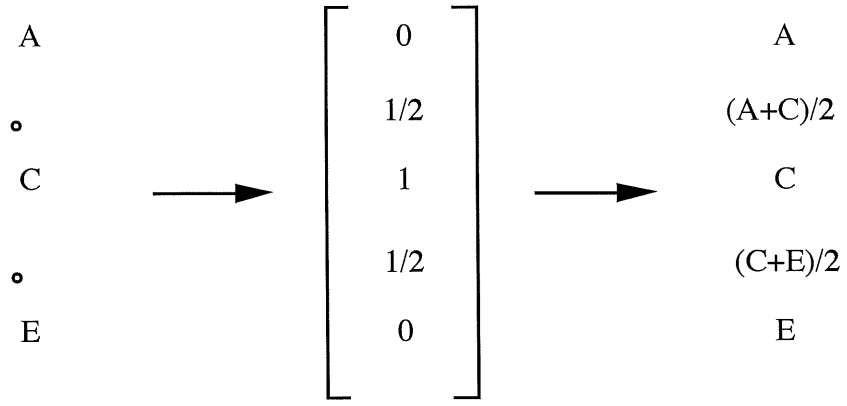
Champ pair Filtre d'ordre zéro Image résultante

Figure 2.2 L'interpolation par un filtre vertical d'ordre 0

L'image résultante de cette interpolation contient des défauts d'«aliasing» (effet de repliement) sur le contour des objets.

2.1.1.2 Interpolation par un filtre vertical d'ordre 1 [15]

Ici, on calcule les moyennes de deux pixels de deux lignes adjacentes pour obtenir les valeurs des niveaux de gris du pixel de la ligne manquante, comme le montre la figure 2.3.



Champ pair Filtre d'ordre 1 Image résultante

Figure 2.3 L'interpolation par un filtre vertical d'ordre 1

Cette approche réduit relativement l'effet d'«aliasing» mentionné précédement mais, par contre, elle produit une image visiblement plus floue.

2.1.1.3 Interpolation par un filtre vertical d'ordre supérieur

Dans ce genre de filtre, on essaie de bien filtrer la haute fréquence verticale d'image en augmentant l'ordre du filtre vertical. Ici, on prend en compte aussi la contribution des lignes existantes plus ou moins proches de la ligne à interpoler.

La figure 2.4 présente un exemple type de cette catégorie de filtre.

$$\frac{1}{64}\begin{bmatrix} -5 \\ 0 \\ 37 \\ 64 \\ 37 \\ 0 \\ -5 \end{bmatrix}$$

Figure 2.4 Exemple d'un filtre vertical d'ordre 2

Des filtres d'ordre supérieur coûtent assez chers, mais ils ne résolvent pas de façon satisfaisante le problème de flou sur l'image.

2.1.2 Interpolation par un filtre temporel d'ordre 1

Ici, le niveau de gris du pixel considéré est la moyenne de celui des pixels des champs adjacents comme le montre la figure 2.5.

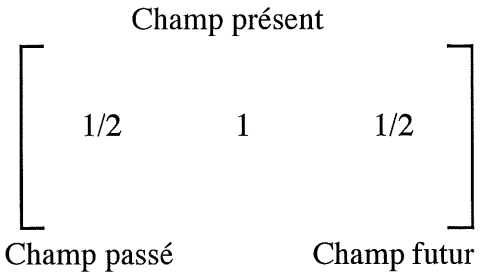


Figure 2.5 Filtre temporel d'ordre 1

Cette technique d'interpolation donne une très bonne résolution verticale mais, par contre elle produit une sérieuse distorsion aux régions en mouvement. Elle est souvent accompagnée par un détecteur de mouvement.

2.1.3 Interpolation par un filtre vertical-temporel (V-T)

À cause de l'entrelacement, l'information utile qui sert à interpoler des lignes manquantes se trouve dans des champs différents, ce qui nécessite l'utilisation d'un filtre vertical-temporel. Ce filtre bidimensionnel fait aussi intervenir les champs passés et les champs futurs du champ à interpoler. La figure 2.6 donne un exemple de ce filtre.

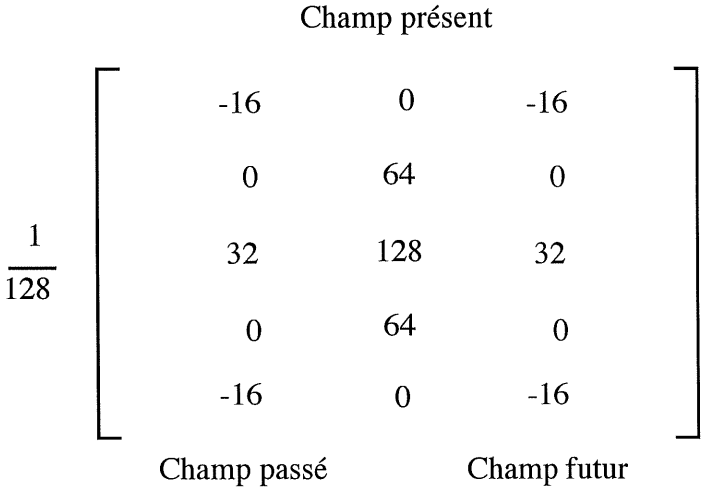


Figure 2.6 Exemple d'un filtre vertical-temporel

Un défaut possible de la technique V-T est la réduction de la résolution des composants spatiaux (verticaux ou diagonaux) à haute fréquence en cas de mouvement. Le filtre V-T sera étudié plus en détail au chapitre 3 et l'image générée par ce filtre sera présentée au chapitre 5.

2.2 Interpolation par filtrage non linéaire

L'interpolation par filtrage non linéaire fait intervenir aussi bien les techniques d'estimation de mouvement et de détection de mouvement que celles de détection de contour. En général, ces techniques ne se trouvent pas en même temps dans un système. La détection de mouvement est probablement la plus simple; sa raison d'être sera présentée à la section 3.2.4. L'estimation de mouvement est probablement la technique la plus complexe. Elle représente certains potentiels appréciables, cependant elle n'est pas nécessairement fiable à cause des mouvements multiples dans l'image.

Dans la partie suivante, nous présentons une revue de la littérature la plus récente de trois méthodes d'interpolation par filtrage non linéaire.

2.2.1 Interpolation directionnelle utilisant les filtres orientables (steerable filters) [7].

Cette approche consiste à utiliser quelques filtres correspondant à certains angles et à interpoler parmi les réponses. On a besoin de savoir combien de filtres sont requis, puis comment interpoler les réponses de façon appropriée. Avec le bon ensemble de filtres et la bonne règle d'interpolation, il est possible de déterminer la réponse d'un filtre dans des directions arbitraires [7]. Le schéma bloc suivant présente bien cette approche.

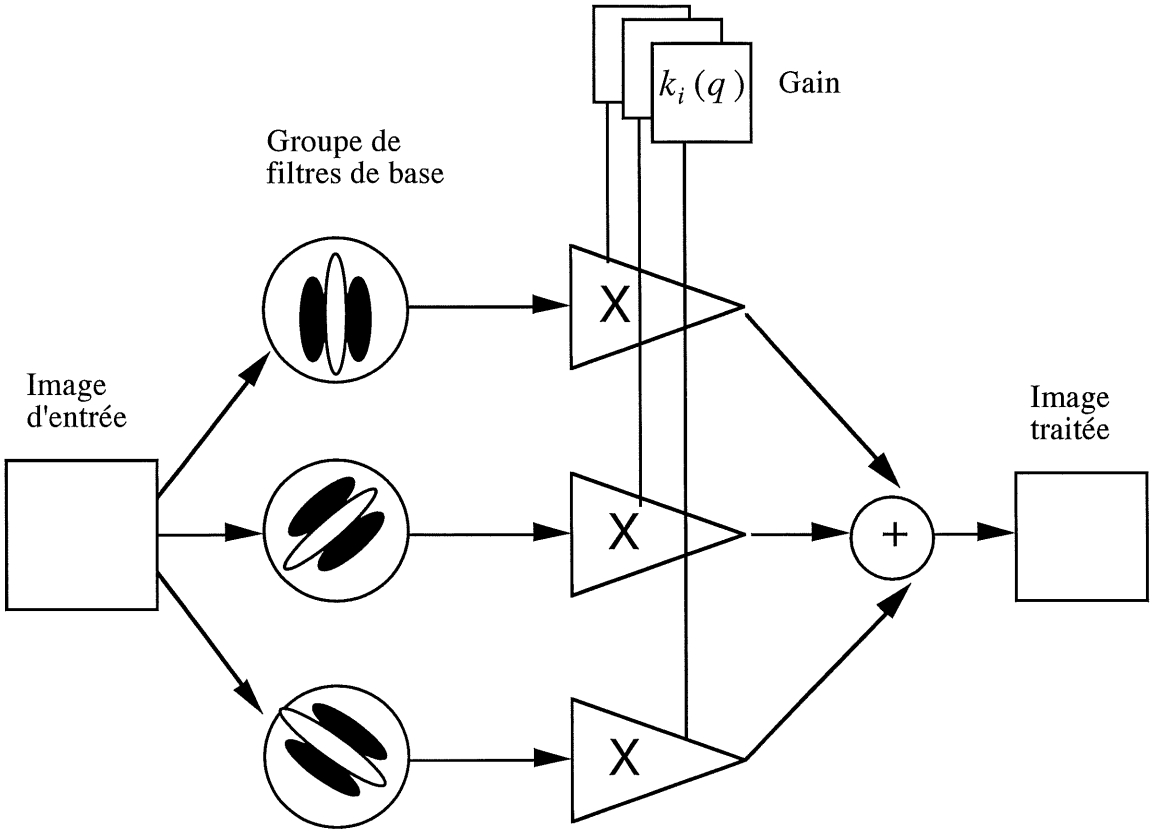


Figure 2.7 Approche d'interpolation utilisant les filtres orientables [5], [6]

Ce type d'interpolation comprend les trois étapes suivantes:

- conception des filtres orientables;
- estimation du mouvement;
- interpolation.

2.2.1.1 Conception des filtres orientables [7]

Toutes les fonctions à bande limitée en fréquence angulaire sont orientables [7]. En pratique, il est avantageux d'avoir moins de filtres de base.

Considérons la fonction bidimensionnelle circulaire gaussienne en coordonnées cartésiennes (x,y)

$$G(x,y)= e^{-(x^2+y^2)} \qquad (2.1)$$

La dérivée directionnelle est bien connue [4], [8], [9], [10], [11]

$$G_2^{0^0} = \frac{\delta^2 e^{-(x^2+y^2)}}{\delta x^2} = (4x^2 - 2)e^{-(x^2+y^2)} \tag{2.2}$$

$$G_2^{60^0} = (16x^2 - 2)e^{-(x^2+y^2)} \tag{2.3}$$

$$G_2^{120^0} = (4y^2 - 2)e^{-(x^2+y^2)} \tag{2.4}$$

D'après la référence [7], il suffit d'avoir trois fonctions de base qui satisfont la relation:

$$\begin{bmatrix} 1 \\ e^{i2\theta} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ e^{i2\theta_1} & e^{i2\theta_2} & e^{i2\theta_3} \end{bmatrix} \begin{bmatrix} k_1(\theta) \\ k_2(\theta) \\ k_3(\theta) \end{bmatrix} \tag{2.5}$$

où les $k_j(\theta)$ sont des fonctions d'interpolation.

En égalant les parties imaginaires et les parties réelles de (2.5), on se trouve face à un système de trois équations à trois inconnues. En choisissant $\theta_1 = 0^0$, $\theta_2 = 60^0$ et $\theta_3 = 120^0$, cela implique

$$k_j(\theta) = \frac{1}{3}[1 + 2\cos(2(\theta - \theta_j))] \tag{2.6}$$

Et finalement, on a:

$$G_2^\theta = k_1(\theta)G_2^{0^0} + k_2(\theta)G_2^{60^0} + k_3(\theta)G_2^{120^0} \tag{2.7}$$

H_2^θ est trouvé en faisant une transformée d'Hilbert de G_2^θ.

G_2^θ et H_2^θ forment deux groupes en quadrature de fonctions de base des filtres orientables.

2.2.1.2 Estimation de l'orientation locale de contour [12]

L'orientation du contour le long d'une direction perpendiculaire θ est estimée par le carré d'une paire de filtres passe-bande dirigée vers l'angle θ. Si on dénote le premier filtre de la paire G^{θ} et le second H^{θ}, «Oriented energy» est défini comme:

$$E(\theta) = [G^{\theta}]^2 - [H^{\theta}]^2 \tag{2.8}$$

En simplifiant équation (2.8) par une série de Fourier, il ne reste que les fréquences paires à cause de l'opération carrée.

$$E(\theta) = C_1 + C_2 \cos 2\theta + C_3 \sin 2\theta + C_4 \cos 4\theta + \cdots \tag{2.9}$$

L'orientation du filtre correspond à la réponse maximale du filtre. Elle est calculée par les termes de plus basse fréquence.

$$\theta_{filtre} = \frac{1}{2}\tan(C_3 / C_2) \tag{2.10}$$

L'orientation locale du contour est perpendiculaire à l'orientation du filtre.

$$\theta_{contour} = \theta_{filtre} + \pi / 2 \tag{2.11}$$

2.2.2.3 Interpolation directionnelle [12]

L'intensité du pixel X montré à la figure 2.8 est interpolée de façon directionnelle comme l'expression suivante:

$$I_X = \frac{1}{2}(I_A + I_B) \tag{2.12}$$

où I_A et I_B sont des intensités du point A et du point B respectivement, dans lesquelles on a interpolé horizontalement avec l'interpolation cubique de Key [13].

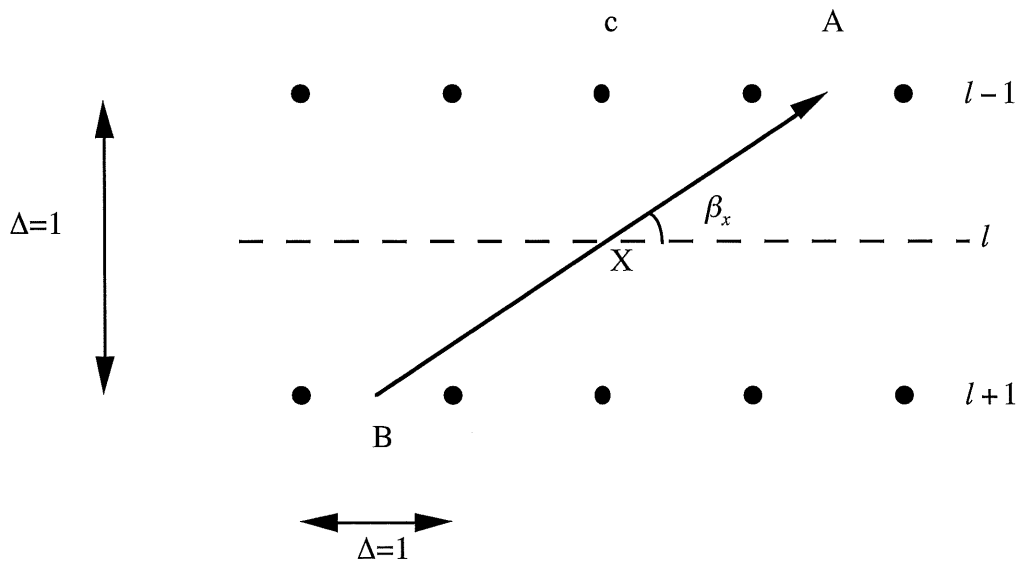


Figure 2.8 L'orientation locale du contour

Pour éviter les intersections et résoudre les problèmes de coin, on fait la comparaison suivante:

$$D_{verticale} = \sum_{i=c-1}^{c+1} |I(i,l-1) - I(i,l+1)| \tag{2.13}$$

$$D_{directionnelle} = \sum_{i=c-1}^{c+1} |I(i+d(\beta_x),l-1) - I(i-d(\beta_x),l+1)| \tag{2.14}$$

$$\text{où} \quad d(\beta_x) = \frac{0.5}{\tan\beta_x} \tag{2.15}$$

Pour chaque position X interpolée comme le montre la figure 2.8:

Si $D_{vertical} < D_{directionnelle}$, I_x est interpolée verticalement comme:

$$I_x = \frac{I(c,l-1) + I(c,l+1)}{2} \tag{2.16}$$

Ailleurs, I_x est interpolée selon la direction grâce à l'équation (2.12).

En somme, une fois les filtres de base sélectionnés, l'interpolation directionnelle utilisant les filtres orientables devient mathématiquement simple à effectuer . Cette interpolation donne un très

bon résultat. Seuls des points le long des lignes et ceux sur des contours horizontaux dans les zones de hautes fréquences spatiales nécessitent une interpolation avec la compensation de mouvement [12].

2.2.2 Interpolation adaptative selon la détection du mouvement et l'orientation des contours [14]

Récemment, Simonetti [14] a proposé un algorithme basé sur un détecteur de mouvement simple et des filtres spatiaux adaptés à la direction de la plus forte corrélation.

D'abord, on cherche à détecter le mouvement. S'il n'y en a pas, on calcule la moyenne du champ passé et du champ futur pour le pixel à interpoler, tel que décrit à la section 2.1.3. La figure 2.9 montre les positions des pixels qui interviennent dans le calcul du pixel manquant.

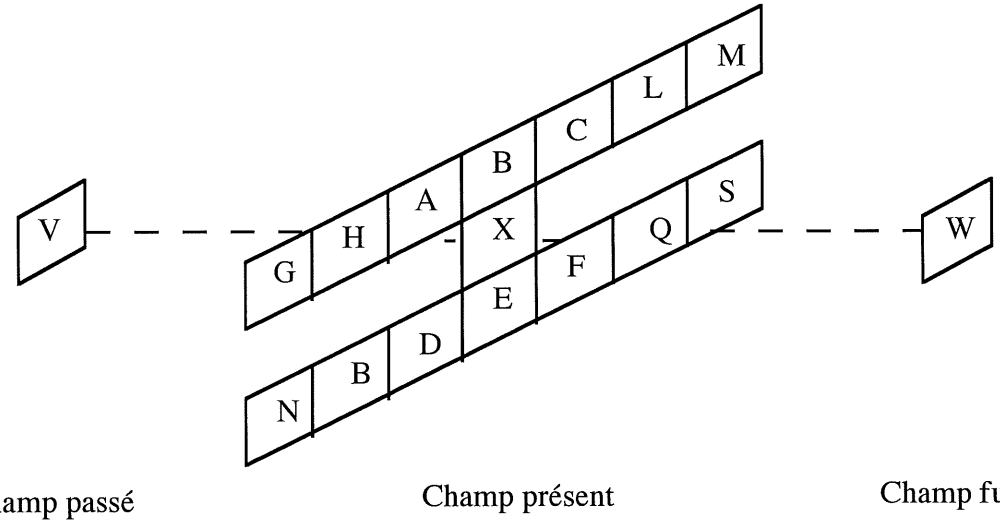


Figure 2.9 Les pixels intervenant dans l'interpolation du pixel X

Plus précisément, le pixel manquant est évalué par l'expression suivante:

$$\text{si } \left[(|V-W|<h_1)\,et(|\frac{V+W}{2}-\frac{B+E}{2}|)<h_2)\right]$$

$$\text{alors } X=(V+W)/2 \qquad (2.17)$$

où h_1 et h_2 sont des seuils de détection. Le premier est plutôt bas (typiquement 16) afin de vérifier si le même pixel d'un objet se trouve dans deux champs différents, tandis que le deuxième est plutôt grand (environ 40) et utilisé pour contrôler la corrélation entre les pixels du même champ.

S'il y a du mouvement, il faudra les filtres spatiaux et seul le champ présent sera considéré. Dans ce cas, il sera important de tenir compte des contours en faisant une interpolation dans la direction de forte corrélation des données de luminance afin de conserver une bonne résolution spatiale.

Les corrélations sont prises dans une petite fenêtre (3x3) centrée sur X: une corrélation significative est dite existante entre deux pixels lorsque la différence absolue entre deux teintes de gris prend une valeur plus basse que le seuil h_2. La présence d'une corrélation peut indiquer la présence d'un contour ou d'une ligne; dans ce cas, une interpolation linéaire est effectuée dans sa direction. Cependant, à cause de la petite fenêtre, seules les pentes qui ont 45^0 ou plus sont prises en compte. Lorsque la présence d'un contour ou d'une ligne fine avec une pente de moins de 45^0 est soupçonnée, l'analyse doit être étendue à une fenêtre 7x3 toujours centrée sur X; aussi dans ce cas, une interpolation linéaire doit-elle être effectuée dans la direction du contour trouvée ou de la ligne considérée. Un mode de réserve (fall-back), dans lequel un simple filtre vertical «Finite Impulsion Response» (FIR) est utilisé (c'est-à-dire, X=((B+E)/2), est aussi prévu pour les cas critiques. Dans ce qui suit, l'algorithme spatial utilisé en cas de mouvement est décrit en détail.

D'abord, une fenêtre 3x3 est considérée, à savoir les pixels A, B, C, D, E et F, et plusieurs situations sont vérifiées.

. La présence d'une frontière verticale ou d'une ligne est détectée quand les couples BE et AD ou BE et CF sont corrélés; dans ce cas, un filtre vertical est utilisé. C'est-à-dire:

$$\text{si } [(|B-E|<h_2)\text{et}(|A-D<h_2)]$$
$$\text{alors } X = (B + E)/2 \qquad (2.18)$$

Dans ce cas, en principe, seule la corrélation entre les pixels B et E peut être vérifiée. Cependant, il faut noter que deux couples dans la même orientation, au lieu d'un seul, augmentent l'efficacité de la détection par rapport au bruit dans l'image. Cette approche est aussi suivie, lorsque c'est possible, pour le reste de l'algorithme.

. Une frontière avec une inclinaison de 64° par rapport à l'horizontale est supposée présente si les couples BD et CE (ou symétriquement, AE et BF) sont corrélés; dans ce cas, X = (B + C + D + E)/4. Ce cas s'applique aussi à de vrais contours mobiles lisses (smooth moving edges); l'aspect lisse (smoothness) pourrait être faible par exemple à cause de l'effet d'intégration temporelle présent dans les caméras vidéos.

Alors, les couples AF, BE, et CD sont pris en compte dans l'ordre pour détecter la présence de contours ou lignes moins fortes. Le nombre de couples corrélés est évalué; alors on procède de la manière suivante:

. Corrélation zéro: on s'attend à ce qu'un contour avec une pente de moins de 45^0 soit présent; une situation possible est montrée à la figure 2.10.

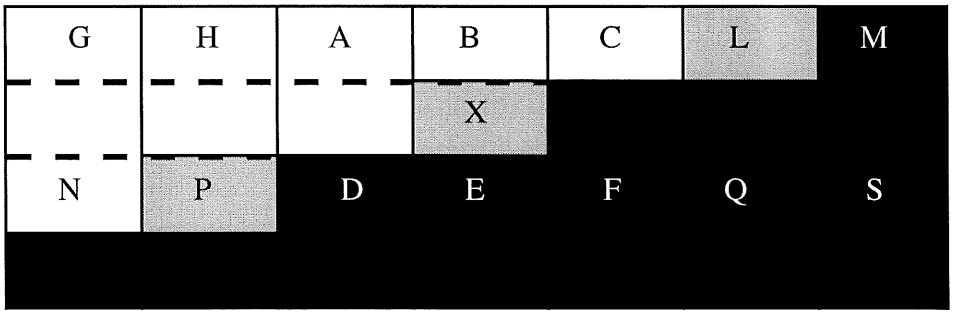


Figure 2.10 Exemple d'un contour flou d'une direction plus petite que 45°

Dans ce cas, l'analyse se fait sur une fenêtre de 7x3, et plusieurs corrélations reliées aux pixels G, H, L, M, N, P, Q et S sont évaluées. Si un couple corrélé est repéré, alors une interpolation est effectuée dans cette direction. Plus précisément, on peut considérer les cas suivants (ensemble avec les cas symétriques):

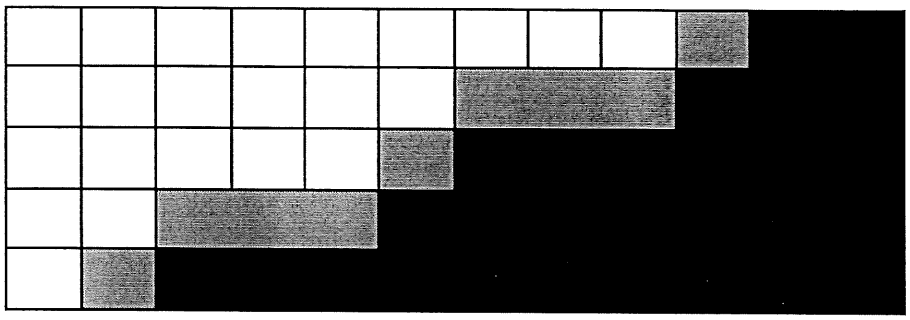
-si LD et CP sont corrélés, alors la pente est de 34° et X = (C + P + D + L)/4;

-si LP et MD ou LP et CN sont corrélés, alors la pente est de 27° et X =(L + P)/2 (ce cas est montré dans la figure 2.11)

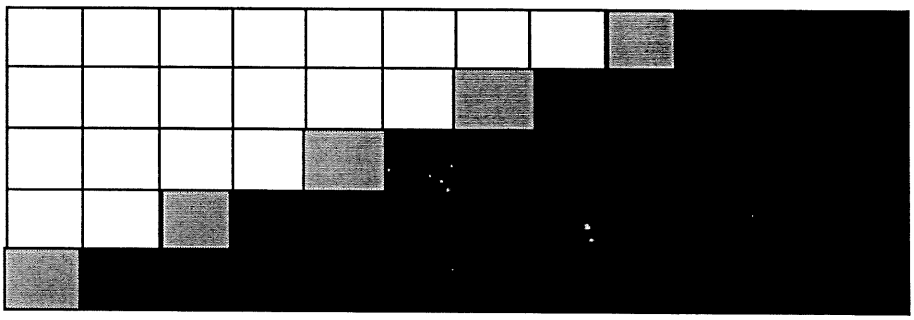
-si MP et LN sont corrélés, alors la pente est de 22° et X = (L + M + N + P)/4

-si MN est corrélé, alors la pente est de 18° et X = (M + N)/2

Si plusieurs couples sont corrélés, on choisit la direction qui montre la plus haute corrélation



Contour reconstruit par un simple filtre vertical



Contour reconstruit par le filtre adaptatif

Figure 2.11 Exemple de la réduction d'effet d'escalier

Il faut noter que cette solution évite l'effet d'escalier (serration effect) qui, autrement, serait présent si on utilisait un filtre vertical sur des contours ayant une pente plus petite que 45° (Figure 2.11). Cependant, à cause de la dimension horizontale finie de la fenêtre (7 pixels), l'algorithme n'est pas capable de reconstruire correctement les contours avec une pente de moins de 18°. Pour de tels contours, le mode «fall-back» est utilisé. Dans ce cas, l'effet d'escalier peut être présent, et on observe:

i) la présence d'une corrélation: un contour ayant un angle de 90^0 (BE) ou 45^0 (AF ou CD) par rapport à l'horizontale a été détecté. Dans ce cas, une interpolation est faite dans cette direction; par exemple, si le couple corrélé est AF, alors X = (A + F)/2.

ii) la présence de deux corrélations: ceci est une situation inhabituelle, qui peut être rencontrée lorsqu'on traite des pixels qui appartiennent aux détails très fins d'une image. Dans ce cas, le mode «fall-back» est utilisé.

iii) la présence de trois corrélations: ici, il est fort probable qu'une ligne a été rencontrée, ainsi les pixels de la ligne et du fond (background) se trouvent-il corrélés. Le cas d'une ligne blanche floue sur un fond noir est montré à la figure 2.12. Il peut arriver qu'aucune information sur la couleur et la pente de la ligne ne soit obtenue de l'analyse de la fenêtre 3x3. Dans ce cas, on devrait poursuivre l'analyse avec une fenêtre de 7x3.

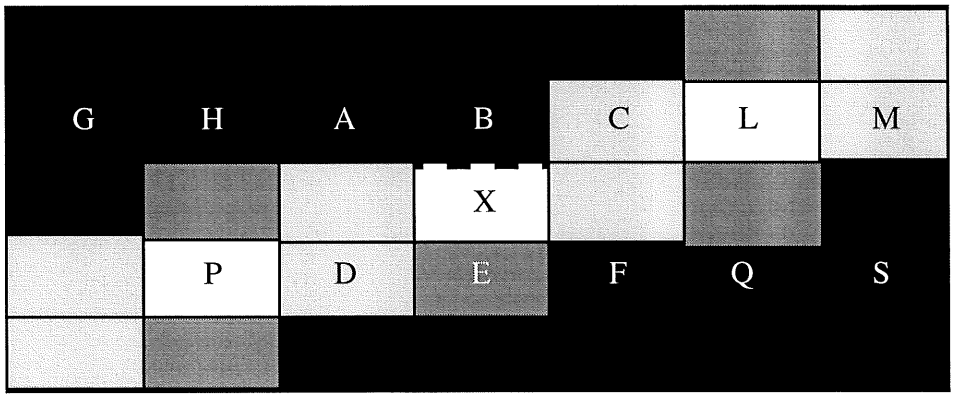


Figure 2.12 Exemple d'une ligne fine floue avec une pente plus petite que 45°

Premièrement, l'arrière-plan (background) doit être discriminé par rapport à la ligne. Si la plus basse rangée de pixels est considérée dans la figure 2.12, on peut voir qu'il est suffisant de considérer les deux triplets sur chacun des côtés: les trois pixels avec lesquels ils ont le plus d'uniformité dans leurs teintes de gris vont appartenir au «background». Par exemple, en se référant à la figure 2.12, on a:

$$|F - Q| + |Q - S| < h_2 < |N - P| + |P - D| \quad (2.19)$$

Une fois le «background» connu, l'orientation sommaire de la ligne est aussi identifiée. Pour la figure 2.12, elle va du coin inférieur gauche vers le coin supérieur droit. Puis, pour reconstruire X avec précision, la pente de la ligne doit être évaluée, en tenant toujours compte des couples de pixels en position par rapport à X. Si $|B - P| < h_2$ la pente est de 45° (le couple CD est déjà connu pour être corrélé); alors X=(C+ D)/2. Autrement, la même analyse utilisée pour la reconstruction d'un contour est faite ici, et les quatre pentes (34°, 27°, 22° et 18°) sont considérées.

Notons que, dans le cas particulier d'une interpolation à ligne fine (thin line interpolation), il est opportun de mettre en marche un procédé de filtrage à posteriori (post-filtering process), afin d'éviter des erreurs qui seraient très visibles pour le spectateur. L'idée est de rendre l'image plus lisse (smooth) dans les cas où un grand contraste est présenté sur des pixels rapprochés, lorsqu'il y a possibilité d'une mauvaise reconstruction. Le pixel manquant est modifié de la façon suivante:

$$X = (B + 2X + E)/4 \quad (2.20)$$

2.2.3 Interpolation avec estimation de mouvement [15]

Martinez a proposé d'utiliser la notion d'estimation de mouvement dans l'interpolation. Cette idée a aussi été abordé par Isnardi [3], [16]. Elle est basée sur un modèle de décalage de lignes dont le schéma bloc se trouve à la figure 2.13.

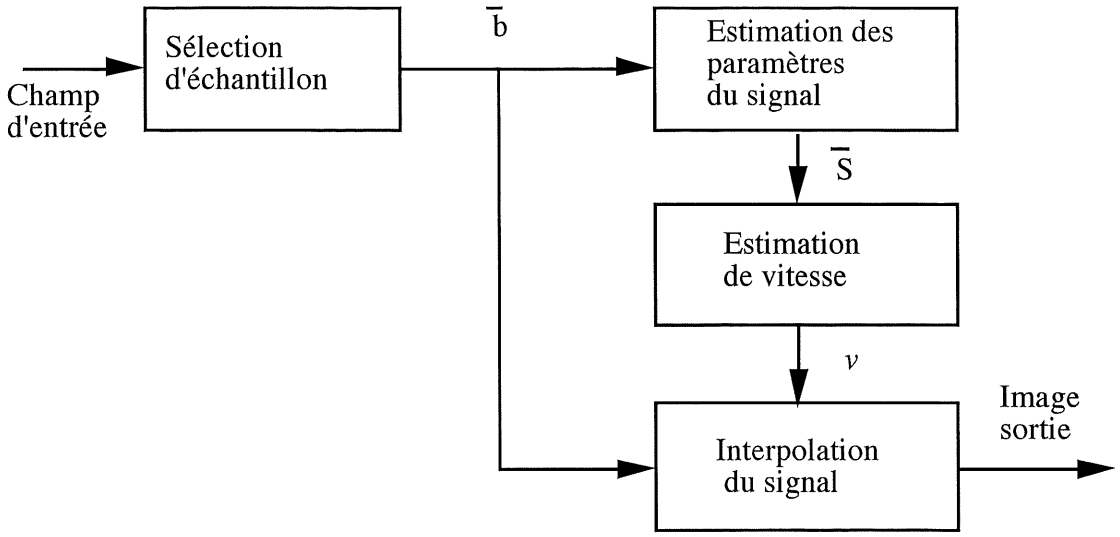


Figure 2.13 Schéma de système d'interpolation par estimation de mouvement [15]

2.2.3.1 Modèle de décalage de lignes [15]

Considérons une petite région d'image autour du point (x_0, y_0) qu'on veut interpoler. Si la région est suffisamment petite, on peut considérer que les lignes adjacentes sont liées approximativement par un simple décalage horizontal. Ce décalage peut être exprimé par une vitesse horizontale v en unité de pixels/ligne. Soit s(x,y), l'intensité de luminance d'image; la relation entre les lignes adjacentes peuvent être représentées par le modèle suivant:

$$s(x,y) = s_0(x - v(y - y_0)) \tag{2.21}$$

Dans cette expression, $s_0(x)$ est l'intensité d'image le long de la ligne à interpoler. Ce modèle est valide seulement dans la petite région autour du point (x, y_0) pour une vitesse constante v. Si la dérivée première partielle de s(x,y) existe, elle peut être écrite sous la forme:

$$v\frac{\delta s}{\delta x} - \frac{\delta s}{\delta y} = 0 \tag{2.22}$$

L'algorithme d'interpolation est basé sur l'équation (2.22). Il implique deux étapes:

i) Une estimation de vitesse v à partir de la fenêtre d'image autour du point considéré.

ii) La vitesse estimée est projettée sur deux lignes adjacentes du point considéré. L'intensité du point considéré est la moyenne de deux points correspondant aux lignes adjacentes.

Ce processus est illustré à la figure 2.14 où v est quantifiée en un nombre entier de pixels par ligne.

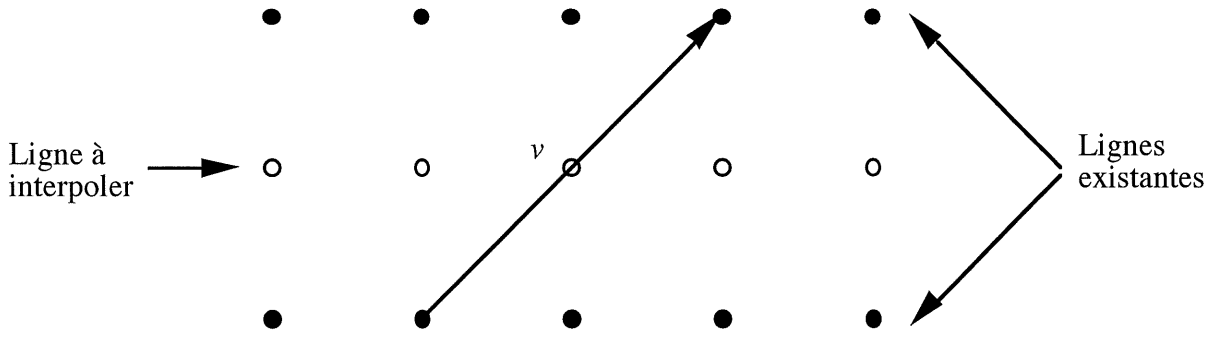


Figure 2.14 Interpolation avec la vitesse estimée v

2.2.3.2 Estimation de vitesse par le modèle paramétrique [15]

Supposons que l'on puisse évaluer les dérivées partielles d'image à un groupe de N points P_i pour i=1...N autour du point considéré dont on veut estimer la vitesse. La technique des moindres carrés qui est introduite dans l'équation (2.22) pour estimer v donne:

$$\min_{v}\left\{\sum_{i=1}^{N}\left(v\frac{\delta s}{\delta x}\Big|_{P_i}+\frac{\delta s}{\delta y}\Big|_{P_i}\right)\right\} \tag{2.23}$$

$$v=\frac{-\sum_{i=1}^{N}\left(\frac{\delta s}{\delta x}\Big|_{P_i}\right)\left(\frac{\delta s}{\delta y}\Big|_{P_i}\right)}{\sum_{i=1}^{N}\left(\frac{\delta s}{\delta x}\Big|_{P_i}\right)^{2}} \tag{2.24}$$

Quand le dénominateur est zéro, le gradient d'image le long de la direction x est zéro et l'estimation de vitesse est impossible. Dans ce cas, on projette simplement une vitesse nulle.

Pour trouver la vitesse estimée de l'équation (2.24), il faut d'abord calculer le gradient d'image à un groupe de points autour du point considéré. Il existe plusieurs méthodes pour déterminer le gradient d'image. Ici, on utilise la méthode qui est basée sur un modèle paramétrique avec une relation linéaire entre les paramètres et le signal. Ce modèle est appliqué seulement sur une petite région d'image autour du point à estimer. Il conduit à la sommation suivante:

$$s(x,y) \approx \bar{s}(x,y) = \sum_{j=1}^{P} S_j \phi_j(x,y) \tag{2.25}$$

Les fonctions $\phi_j(x,y)$ pour j=1...P sont des fonctions bidimensionnelles de base que l'on peut choisir arbitrairement et les $\{S_j\}$ sont des paramètres du modèle.

Les échantillons du champ donné sont utilisés pour estimer les paramètres $\{S_j\}$ et les gradients sont calculés en différenciant $\bar{s}(x,y)$. Pour avoir un algorithme robuste en présence de bruit, on utilise un modèle surdéterminé, c'est-à-dire qu'il y a plus d'échantillons de signal que de paramètres.

Pour chaque point où la vitesse est à estimer, une fenêtre de M échantillons est extraite. En pratique, on utilise une grille de dix échantillons de deux lignes adjacentes (M=10). Les paramètres sont déterminés en minimisant le carré de l'erreur entre le modèle et les échantillons traités. Étant donné la linéarité du modèle, l'estimation des paramètres se fait en résolvant l'équation suivante:

$$\min_{\{S_j\}} \left\{ \sum_{k=1}^{M} \left(s(x_k,y_k) - \sum_{j=1}^{P} S_j \phi_j(x_k,y_k) \right)^2 \right\} \tag{2.26}$$

$$\Rightarrow \bar{S} = (A^T A)^{-1} A^T \bar{b} \tag{2.27}$$

où

$$A = \begin{bmatrix} \phi_1(x_1,y_1) & \cdots & \phi_P(x_1,y_1) \\ \phi_1(x_M,y_M) & \cdots & \phi_P(x_M,y_M) \end{bmatrix} \tag{2.28}$$

$$\bar{b} = \begin{bmatrix} s(x_1,y_1) \\ \vdots \\ s(x_M,y_M) \end{bmatrix} \text{ et } \bar{S} = \begin{bmatrix} S_1 \\ \vdots \\ S_P \end{bmatrix} \tag{2.29}$$

Avec les fonctions polynôminales de base (P=5)

$$\phi_1(x,y)=1 \quad \phi_2(x,y)=x \quad \phi_3(x,y)=y \quad \phi_4(x,y)=x^2 \quad \phi_5(x,y)=xy$$

La sélection a tenu compte des facteurs suivants:

i) la très petite dimension de la région du modèle de l'image.

ii) la surdétermination du modèle. Ici on ne cherche pas à avoir une représentation exacte de la réalité en présence de bruit dans l'image.

iii) les échantillons sont limités à une petite région, donc les polynômes constituent un meilleur choix pour les fonctions de base.

Une fois les paramètres $\{S_j\}$ trouvés, l'équation (2.24) est réduite à:

$$v = \frac{-\bar{S}^T (G_x^T G_y)\bar{S}}{\left|G_x \bar{S}\right|^2} \tag{2.30}$$

$$G_x = \begin{bmatrix} \frac{\delta\phi_1}{\delta x}\Big|_{P_1} & \cdots & \frac{\delta\phi_P}{\delta x}\Big|_{P_1} \\ \vdots & & \vdots \\ \frac{\delta\phi_1}{\delta x}\Big|_{P_N} & \cdots & \frac{\delta\phi_P}{\delta x}\Big|_{P_N} \end{bmatrix} \tag{2.31}$$

$$G_y = \begin{bmatrix} \frac{\delta\phi_1}{\delta y}\Big|_{P_1} & \cdots & \frac{\delta\phi_P}{\delta y}\Big|_{P_1} \\ \vdots & & \vdots \\ \frac{\delta\phi_1}{\delta y}\Big|_{P_N} & \cdots & \frac{\delta\phi_P}{\delta y}\Big|_{P_N} \end{bmatrix} \tag{2.32}$$

Donc, cette approche est basée sur deux modèles différents d'images spatiales. Elle produit moins d'«aliasing spatial» et montre des contours beaucoup plus nets que ceux produits par un filtre vertical d'ordre 1.

2.3 Conversion à haute résolution [17]

La conversion à haute résolution se fait dans les conditions optimales, suivant la recommendation no 601-1 du CCIR [18]. La portion vidéo d'une ligne horizontale a une durée de 53,3 μsec, échantillonnée à 13,5MHz, soit 720 échantillons par ligne. La durée d'une ligne de sortie («output») du doubleur de lignes est la moitié de celle du signal vidéo normal. Ainsi, doit-on échantillonner à une fréquence au moins du double de 13,5MHz, soit 27MHz. Verticalement, un champ vidéo a 240 lignes actives qu'on double à 480 dans un cadre d'image. Ceci est présenté dans la figure 2.15-a.

Une image DSC-TVHD (Digital Spectrum Compatible-HDTV) a 720 lignes actives ayant 1280 pixels actifs par ligne. Notons ici qu'il s'agit d'un taux de 16 pixels par 9 lignes, le même taux que l'aspect de l'image, par conséquent, le pixel du DSC-TVHD doit être carré.

La tâche du convertisseur développé ici est de convertir une image entrelacée de 720*480 à une image progressive de 1280*720. Étant donné la différence du taux d'aspect, l'image vidéo existante peut couvrir seulement une partie du format d'image DSC-TVHD, les deux côtés des panneaux resteront noirs.

En termes généraux, le processus de conversion se présente selon la figure 2.15. Premièrement, le nombre de lignes/image est triplé à 1440 par interpolation (figure 2.15b). Ensuite, on le diminue à 720. Les 720 pixels horizontaux par ligne sont emmagasinés et lus dans un temps qu'occupent normalement 960 pixels DSC-TVHD. L'image TVHD résultante a 960*720 pixels actifs, correspondant aux taux original vidéo; 160 pixels resteront noirs sur chaque côté. La décimation résultante est présentée dans à la figure 2.15-c et le taux de conversion à la figure 2.15-d. La figure 2.15-e présente l'écran TVHD avec les côtés vides des panneaux (la partie hachurée).

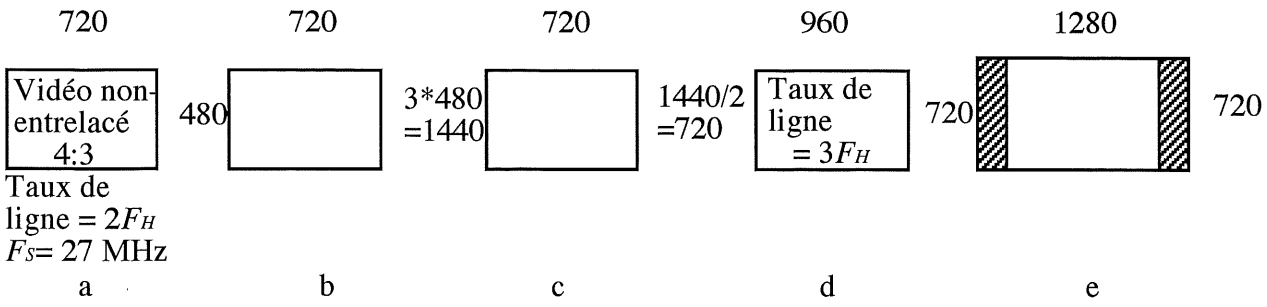


Figure 2.15 Les étapes de surconversion avec l'image vidéo standard

De nos jours, il existe beaucoup de films présentant des taux d'aspect à l'écran DSC-TVHD. Lorsqu'on considère les étapes d'introduction graduelle du DSC-TVHD, il est probable que le taux d'aspect des caméras vidéos soit disponible ou bien que les caméras existantes soient adaptées dès la première étape.

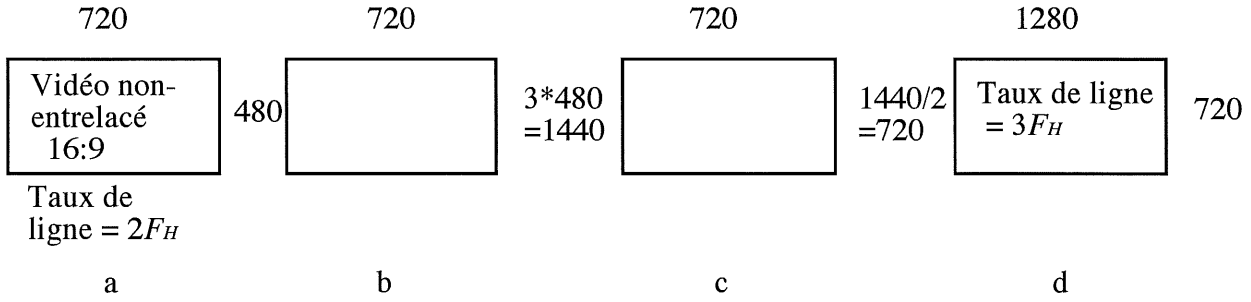


Figure 2.16 Les étapes de surconversion avec la vidéo 16:9

La figure 2.16 présente schématiquement la conversion à partir de la vidéo 16:9. Les figures 2.16-a, 2.16-b, 2.16-c présentent un processus identique à celui des figures 2.15-a, 2.15-b et 2.15-c. Dans la figure 2.16-d, les 720 échantillons par ligne sont emmagasinés comme précédemment à la figure 2.15, mais ici, il sont lus dans un intervalle de temps de 1280 pixels au lieu de 960 dans le cas DSC-TVHD. L'image vidéo agrandie 16:9 occupe tout l'écran TVHD.

2.4 Conclusion

L'interpolation par le filtrage est intéressante à cause de sa simplicité. Avec un filtre bidimensionnel approprié, on peut obtenir une image acceptable à la sortie. Par contre, les très hautes fréquences et les composantes diagonales en mouvement sont souvent mal filtrées, ce qui introduit l'effet d'escalier aux contours inclinés et des ombres fictives sur les contours fins.

L'interpolation non linéaire avec le filtre de compensation de mouvement [19], [20] peut être une solution aux problèmes générés par l'entrelacement du signal vidéo, notamment le papillotement interligne, la ligne grouillante, l'aliasing vertical. Cependant, les techniques d'estimation de mouvement sont très délicates et coûteuses. En général, il est plus économique de détecter seulement le mouvement au récepteur.

La combinaison de l'estimation de l'orientation locale de contour utilisant les filtres orientables avec l'interpolation spatiale directionnelle donne un très bon résultat pour la majorité des images traitées. Par contre, d'une part, il y a encore du flou dans les régions de très hautes fréquences verticales et de la discontinuité sur les contours fins. L'utilisation de détecteur de mouvement simple avec des filtres adaptatifs [14] améliore visiblement la qualité de l'image résultante dans les zones fixes. D'autre part, la détection de contours par les techniques existantes peut être fragile ou parfois non consistante, ce qui produit occasionnellement le phénomène de discontinuité et de papillotement.

Dans le chapitre suivant, nous présentons une configuration du doubleur de lignes qui se caractérise principalement par:

- une détection fiable de contour et
- une interpolation adaptative spatio-temporelle

Nous mettons également l'accent sur le développement et l'utilisation d'algorithmes simples, matériellement réalisables et efficaces.

3

CONCEPTION D'UN DOUBLEUR DE LIGNES

Dans le domaine de traitement d'images, on a toujours observé des défauts inhérents aux images entrelacées. Parmi ces anomalies d'images, on peut citer le manque de netteté de l'image dû notamment à la structure des lignes et le papillotement interligne apparaissant lorsque les lignes paires et les lignes impaires sont suffisamment différentes.

En vue de remédier aux défauts mentionnés ci-dessus, nous présentons dans ce chapitre, le résultat de la conception et du développement d'un nouveau doubleur de lignes qui a pour objectif de convertir le signal entrelacé en signal progressif, tout en permettant de doubler la résolution verticale de l'image sans compromettre sa résolution temporelle.

Ce chapitre portant sur le dé-entrelacement de séquence d'image entrelacée s'articule comme suit. Après une présentation de la configuration du doubleur de lignes proposé, nous en présentons son principe et son fonctionnement de base avec un accent particulier sur le détecteur de contour et sa fortification. Nous développons aussi divers types d'interpolation, à savoir: l'interpolation temporelle, l'interpolation selon le contour, l'interpolation spatio-temporelle. Une attention particulière est accordée au traitement de la détection de mouvement. Le chapitre se termine par une conclusion présentant les mérites et les limites de l'algorithme développé.

3.1 Configuration du doubleur de lignes proposé

Le doubleur de lignes proposé compte plusieurs blocs distincts dont chacun possède une fonction précise dans le système global. Le schéma bloc de ce doubleur de lignes se trouve à la figure 3.1.

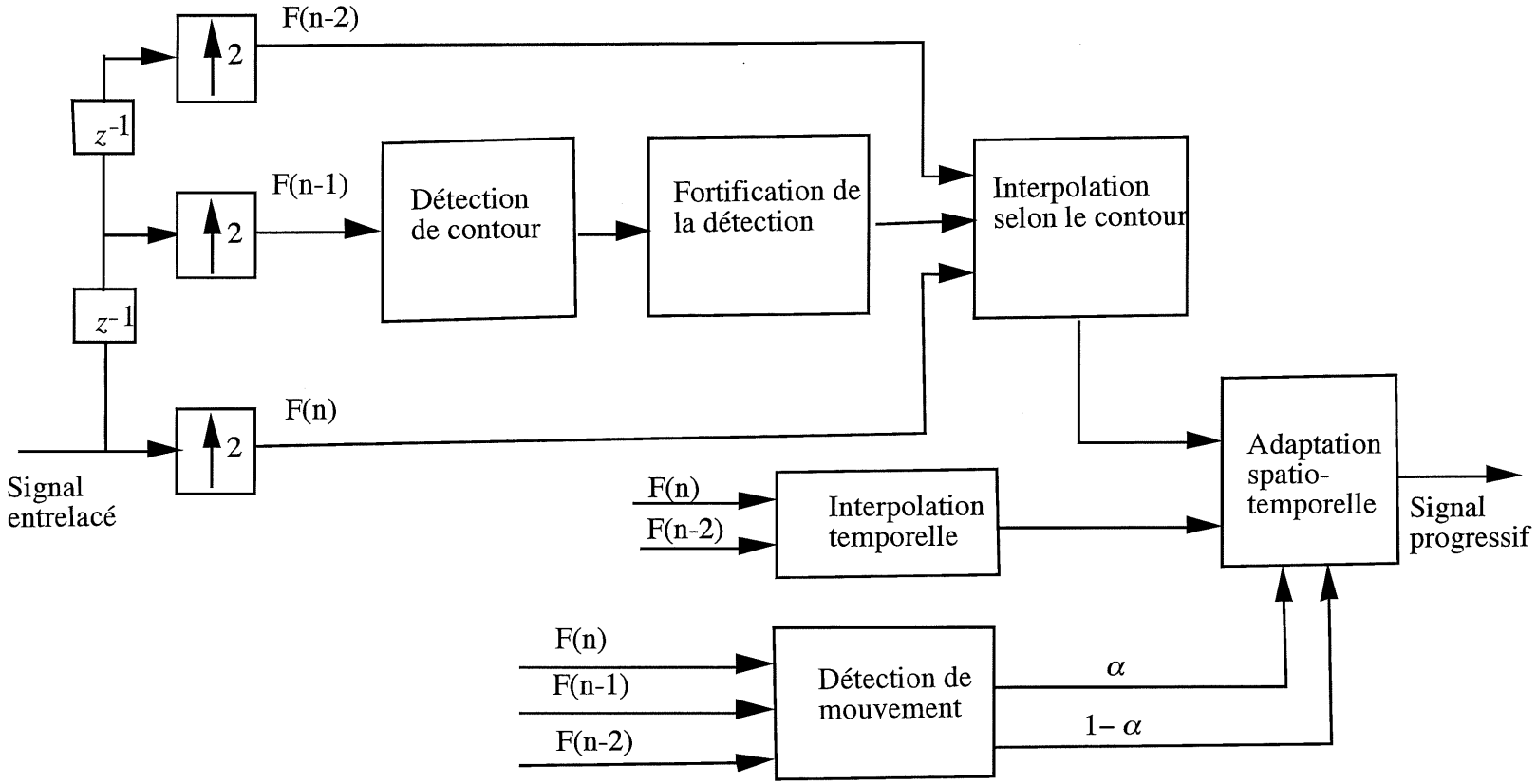


Figure. 3.1 Doubleur de lignes proposé

On voit que ce doubleur de lignes est constitué de six grands blocs: la détection de contour, la fortification de la détection, l'interpolation selon le contour, l'interpolation temporelle, la détection de mouvement et l'adaptation spatio-temporelle.

Comme les défauts sont très visibles sur les contours, ces derniers doivent être détectés et requièrent un traitement spécial. Cette détection est réalisée par le bloc de détection de contour et consolidée par la partie de fortification.

La détection de mouvement a pour tâche de classer les zones dynamiques et statiques dans une image. En fait, l'interpolation temporelle est meilleure dans la zone statique et l'interpolation selon le contour est très bonne pour la zone dynamique ou pour le contour en mouvement.

Enfin, l'adaptation spatio-temporelle calcule la contribution des deux interpolations dans le résultat final selon le degré de mouvement.

3.2 Principe et fonctionnement

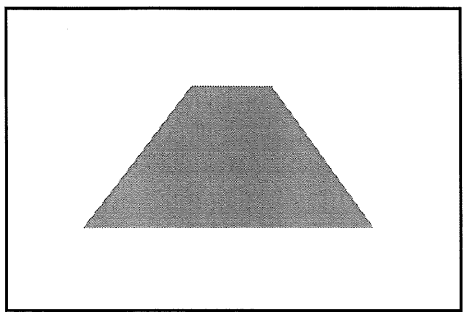
D'abord, l'échantillon du signal entrelacé est suréchantillonné par un facteur 2. Cet échantillon est ensuite traité sur trois champs successifs F(n-2), F(n-1) et F(n).

La détection du contour est réalisée sur le champ présent F(n-1). Elle est rendue robuste et consistante par la technique de renforcement des décisions séquentielles. À la sortie de la partie de fortification de la détection, le signal du champ présent contient l'information des directions de contour détectées. Cette information détermine le choix des filtres V-T adaptatifs. Enfin, à la sortie de ces filtres, on obtient une interpolation spatiale, bonne pour les images dynamiques.

Étant donné que le filtre temporel donne une meilleure interpolation pour l'image statique, le détecteur de mouvement produit deux facteurs α et $1-\alpha$ qui déterminent la contribution des deux interpolations spatiale et temporelle dans le résultat final.

3.2.1 Détection de contour

Dans une image, l'ensemble des changements brusques du niveau de gris forme des contours. Ainsi, pour un point donné, la détection du contour est déterminée comme la direction où la variation du niveau de gris est la plus faible. S'il n'y a pas de différence notoire de variation dans les directions, cela signifie que le point n'est pas sur le contour. Les figures 3.3a et 3.2b donnent un exemple de ce fait.



255	255	255	255	255	255	255
255	255	255	100	255	255	255
255	255	100	100	100	255	255
255	100	100	100	100	100	255
255	255	255	255	255	255	255

blanc = 255
gris = 100
noir = 0

a b

Figure 3.2 Exemple d'un contour incliné: (a) une pyramide (b) son équivalent en valeur de pixel

L'interpolation selon la direction du contour va donner une valeur beaucoup plus exacte que celle qui provient d'une direction quelconque qui peut rendre flou le contour. Sur la base de ces observations, on va concevoir un détecteur de contour suivant les cinq directions statistiquement les plus fréquentes dans l'image. Le schéma de ce détecteur se trouve à la figure 3.3.

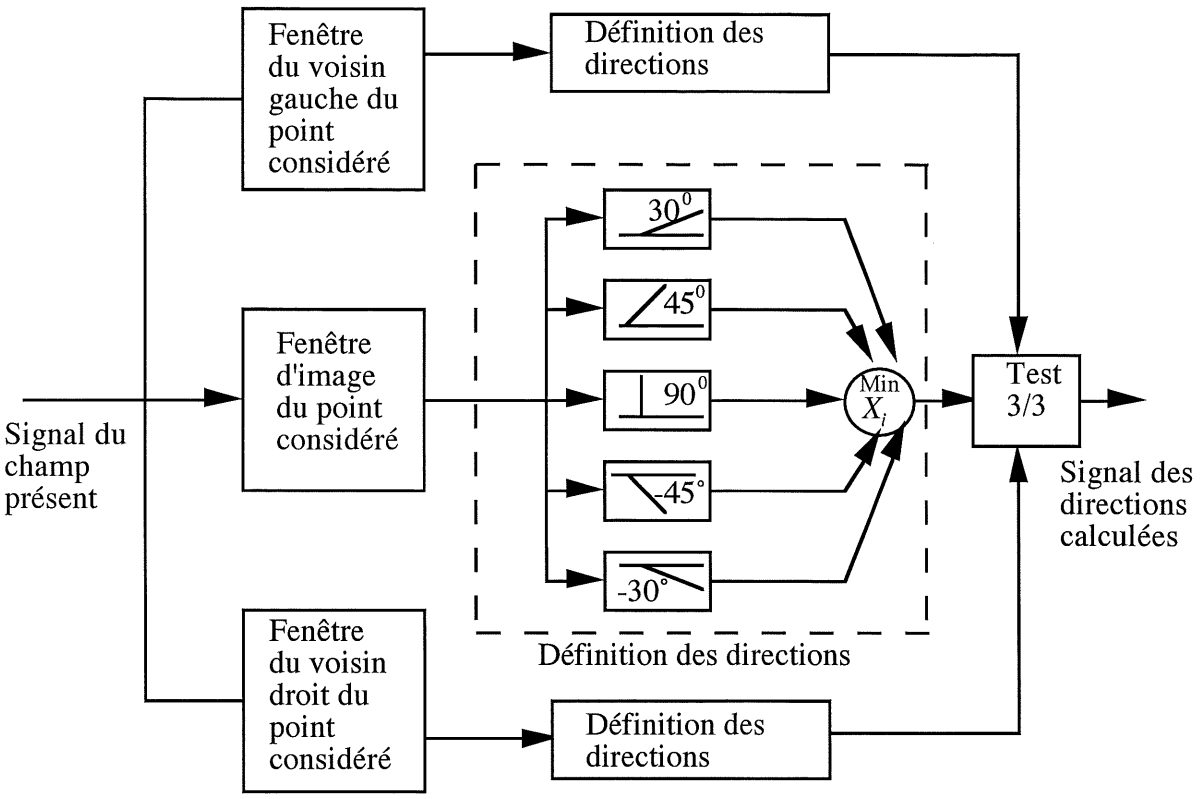


Figure 3.3 Détecteur de contour du doubleur de lignes proposé

Ce détecteur de contour est basé sur des filtres spatiaux verticaux-horizontaux du champ présent qui contient toute l'information nécessaire à la définition du contour. Ces filtres restent invariables dans le temps et ne sont pas influencé par l'effet de mouvement.

Le contour est détecté à partir de la direction dominante, à savoir, la direction où la variation de luminance est la plus faible. La priorité est donnée à la direction verticale (90^0), car cette direction interpole le mieux les points en dehors du contour. La figure 3.4 montre une fenêtre d'image.

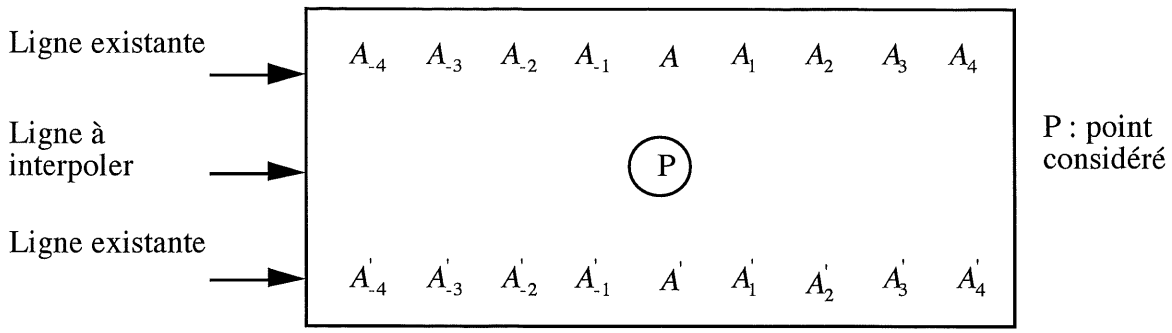


Figure 3.4 Fenêtre d'image du champ présent

La fenêtre d'image contient 18 pixels de 2 lignes existantes autour du point considéré (à déterminer pour la direction du contour). Donc, pour trouver la direction du pixel à interpoler, le détecteur doit emmagasiner l'information de la ligne précédente et celle de la ligne suivant la ligne à interpoler. A_{-4} à A_4 et A'_{-4} à A'_4 présentent des pixels qui vont intervenir dans la détermination de la direction du point P. Pour éviter l'effet des bordures de l'image, notre algorithme considère de plus que l'image traitée est pliée de haut en bas et de gauche à droite.

Pour cette fenêtre, les filtres spatiaux des directions se définissent comme le montre la figure 3.5.

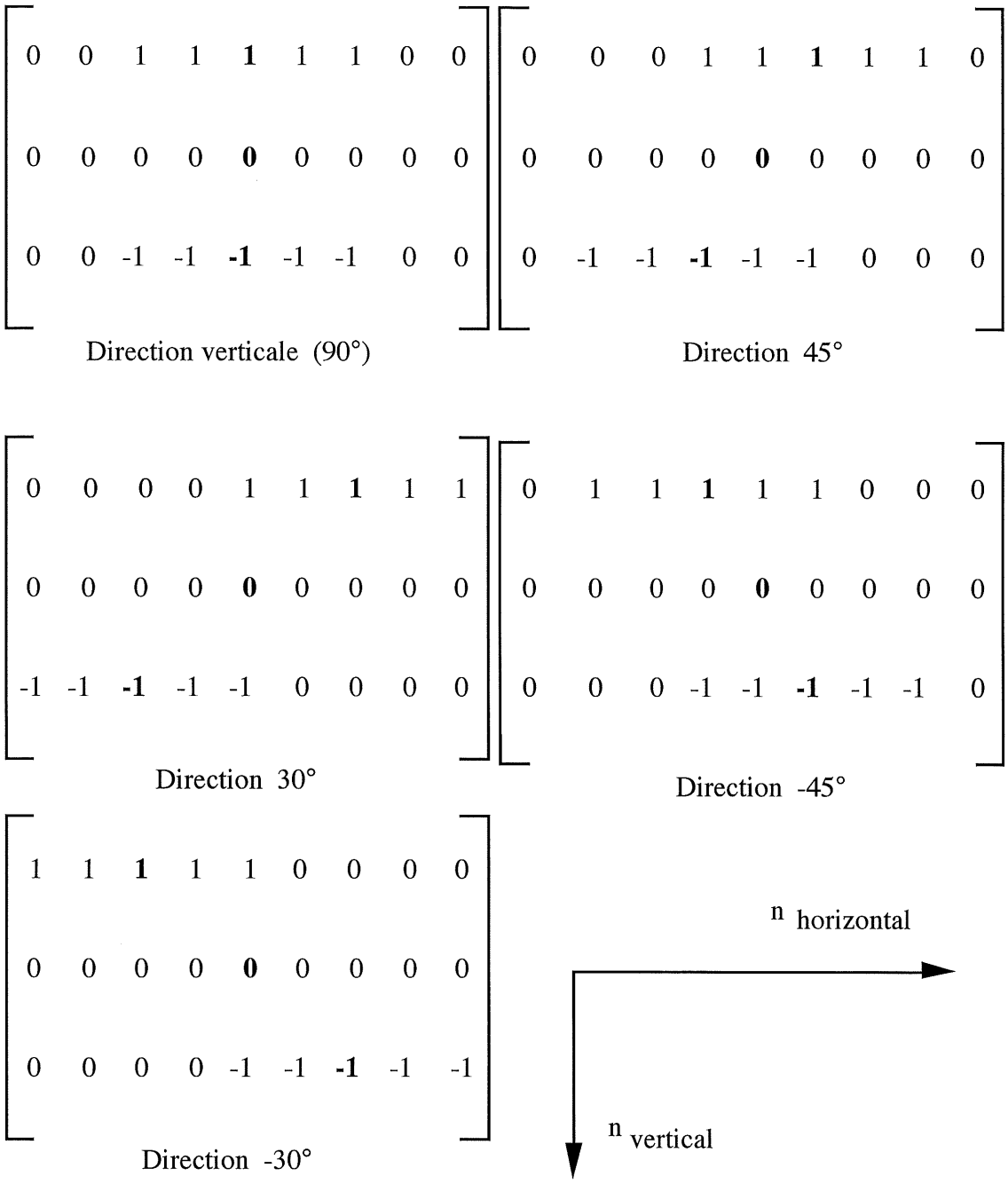


Figure 3.5 Les filtres verticaux-horizontaux du champ présent

Le détecteur de contour conçu détecte les quatre directions statistiquement les plus fréquentes et la direction verticale prédominante.

Algébriquement, on a:

$$X_{90^0} = |(A_{-2} + A_{-1} + A + A_1 + A_2) - (A'_{-2} + A'_{-1} + A' + A'_1 + A'_2)| \quad (3.1)$$

$$X_{45^0} = |(A_{-1} + A + A_1 + A_2 + A_3) - (A'_{-3} + A'_{-2} + A'_{-1} + A' + A'_1)| \quad (3.2)$$

$$X_{30^0} = |(A + A_1 + A_2 + A_3 + A_4) - (A'_{-4} + A'_{-3} + A'_{-2} + A'_{-1} + A')| \quad (3.3)$$

$$X_{-30^0} = |(A_{-4} + A_{-3} + A_{-2} + A_{-1} + A) - (A' + A'_1 + A'_2 + A'_3 + A'_4)| \quad (3.4)$$

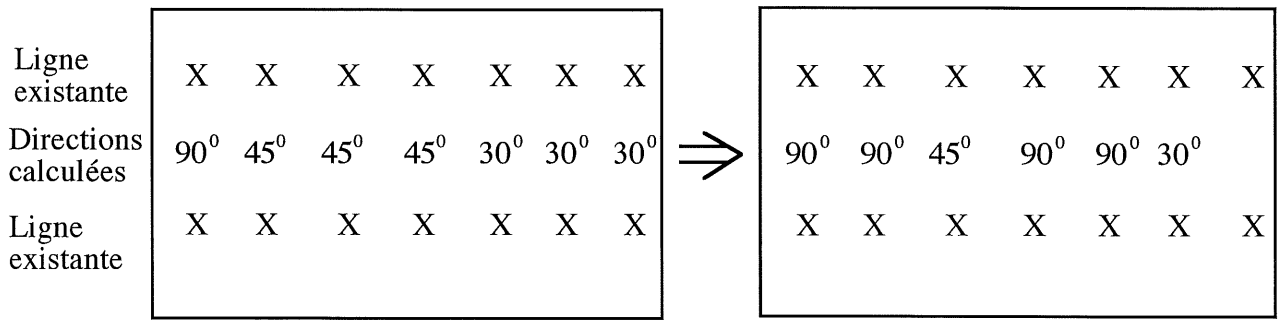
$$X_{-45^0} = |(A_{-3} + A_{-2} + A_{-1} + A + A_1) - (A'_{-1} + A' + A'_1 + A'_2 + A'_3)| \quad (3.5)$$

L'utilisation des blocs de 5 pixels permet d'éviter les fausses détections causées par le bruit.

Les cinq valeurs calculées selon les directions ($X_{90^0}, X_{45^0}, X_{30^0}, X_{-30^0}, X_{-45^0}$) se sont comparées pour identifier la valeur la plus petite. C'est grâce à la direction de cette dernière que le contour est détecté.

$$D_{i;calculé} = Min\{X_i\big|_{i=90^0,45^0,30^0,-30^0,-45^0}\} \quad (3.6)$$

Le processus se répète pour les deux pixels voisins du pixel en question avec la fenêtre d'image décalée à gauche et à droite d'une colonne. Pour ne garder que des contours qui sont bien solides et bien distingués, les directions de ces deux pixels voisins doivent être les mêmes que celle du pixel considéré, afin que la direction de ce pixel soit retenue. C'est un test de 3/3 ou de «majorité absolue». Si le test n'est pas satisfaisant, la direction verticale (90^0) est retenue. La figure 3.6 montre un exemple de ce processus de sélection non récursif.



Avant le test 3/3 Après le test 3/3

Figure 3.6 Exemple de processus de sélection

Ce processus peut être représenté par la relation suivante:

$$D_{retenu}(n_h,n_v)=\begin{cases} D_{i;calculé}(n_h,n_v) & si \quad D_{i;calculé}(n_h-1,n_v)=D_{i;calculé}(n_h+1,n_v)=D_{i;calculé}(n_h,n_v) \quad où \quad i=45°,30°,-30°,-45° \\ D_{90°} & \text{sinon} \end{cases} \tag{3.7}$$

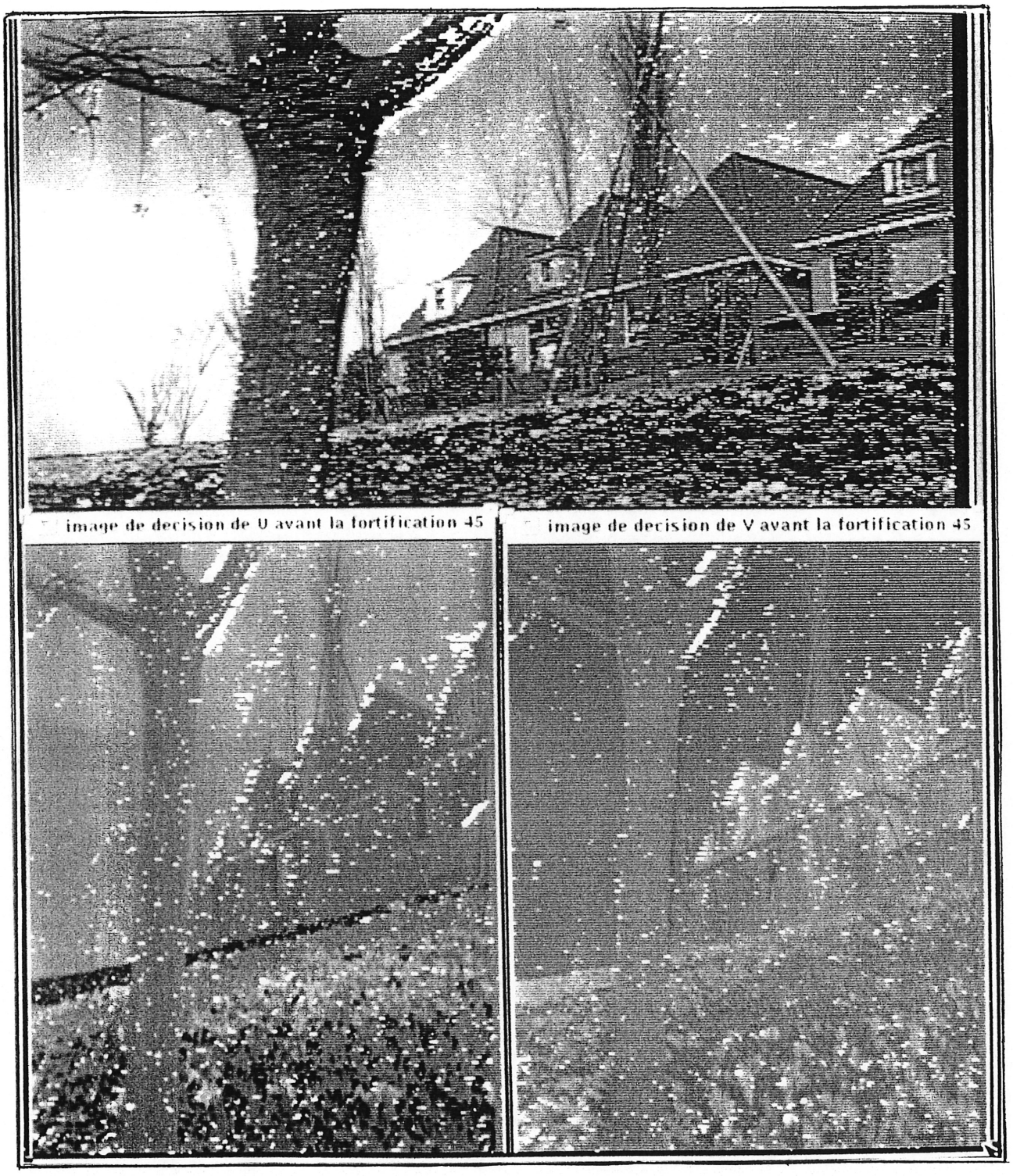
On peut finalement avoir:

$$d_{i;retenu}(n_h,n_v)=\begin{cases} 1 & \text{si} \quad D_{retenu}(n_h,n_v)=D_i(n_h,n_v) \quad où \quad i=45°,30°,-30°,-45° \\ 0 & \text{sinon} \end{cases} \tag{3.8}$$

Les contours détectés d'une image critique sont illustrés par les figures 3.7, 3.8, 3.9, 3.10 pour des directions respectives $45^0,30^0,-30^0,-45^0$. Les contours détectés sont présentés par des points blancs. On voit qu'il y a de la discontinuité sur les contours et il existe beaucoup de points isolés. Ces derniers présentent des faux contours ou des contours fragiles.

La partie de fortification de détection va consolider les vrais contours et éliminer les faux contours.

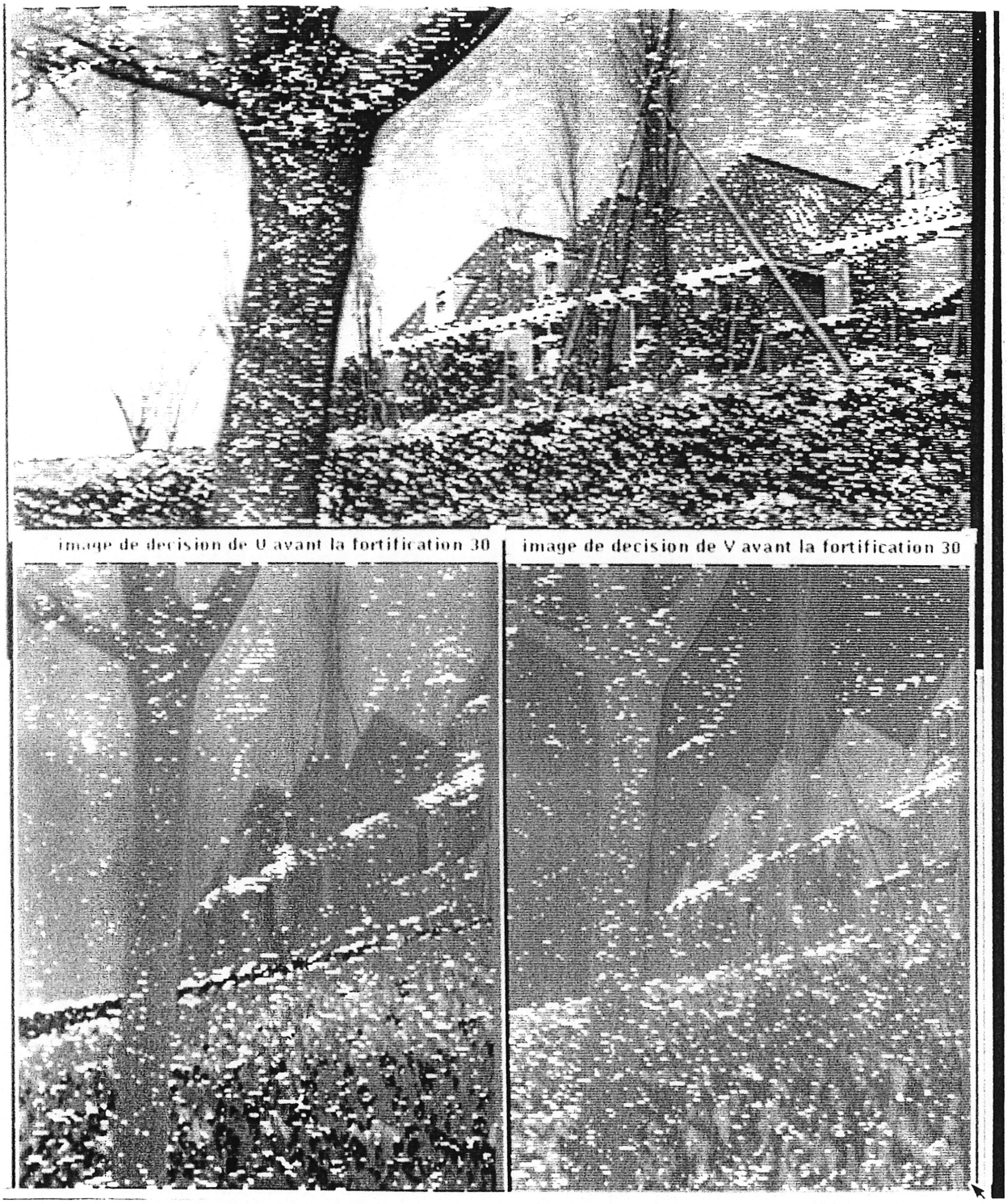
a



b c

Figure 3.7 Image «Flower» avec les contours de 45° détectés avant la fortification
a) la composante Y b) la composante U c) la composante V

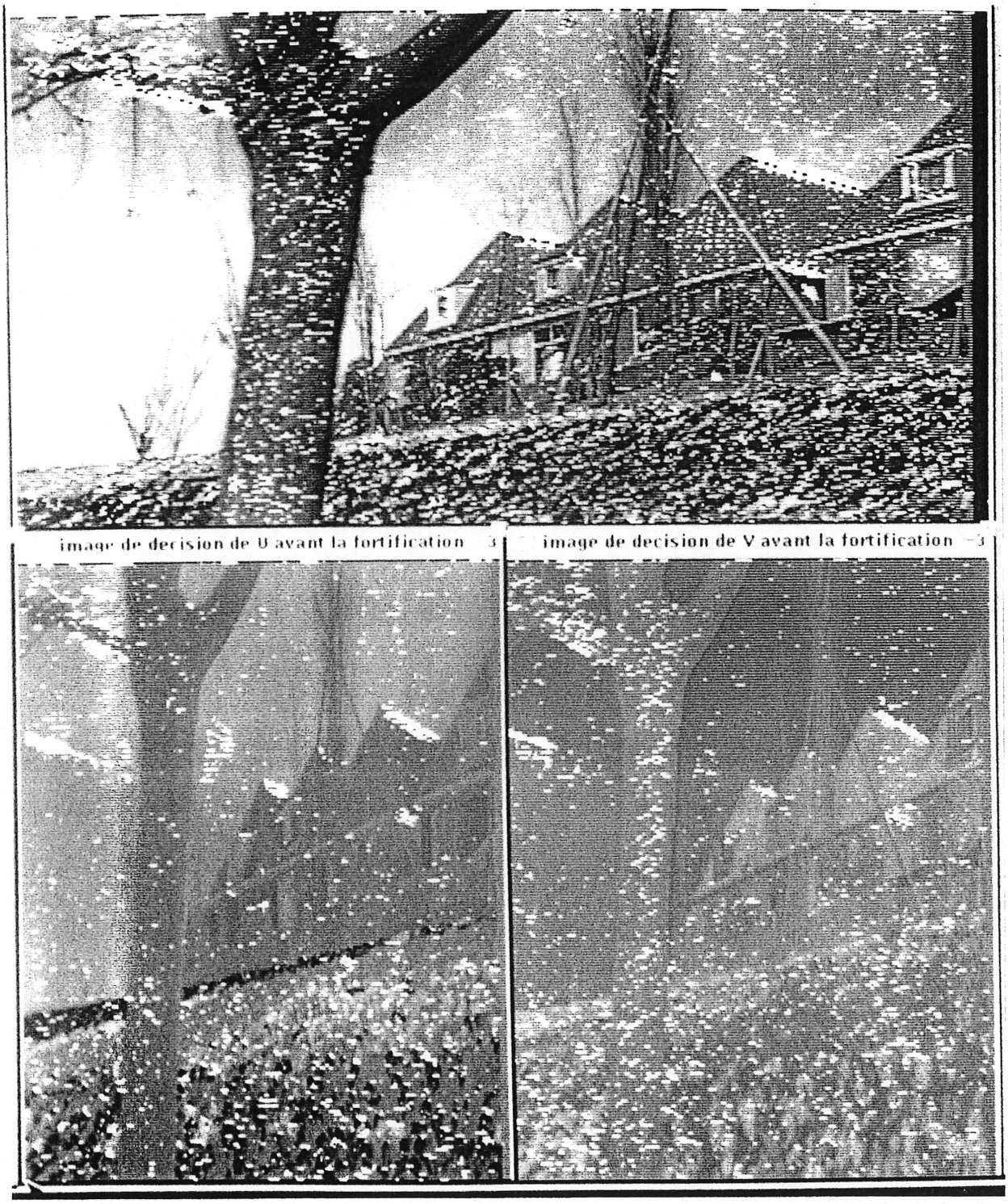
a



b c

Figure 3.8 Image «Flower» avec les contours de 30° détectés avant la fortification
a) la composante Y b) la composante U c) la composante V

a



b c

Figure 3.9 Image «Flower» avec les contours de -30° détectés avant la fortification
a) la composante Y b) la composante U c) la composante V

a

b c

Figure 3.10 Image «Flower» avec les contours de -45° détectés avant la fortification
a) la composante Y b) la composante U c) la composante V

3.2.2 Fortification de la détection

Les directions détectées par le détecteur de contour vont subir une autre opération de filtrage qui a pour effet non seulement d'enlever les discontinuités dans les contours mais aussi d'éliminer les points isolés. Ainsi, les contours fragiles ou les faux contours sont éliminés et les vrais contours sont consolidés. Il est à noter que la direction verticale détectée n'a pas besoin de passer par cette étape. Le schéma de la partie de fortification est présenté à la figure 3.11.

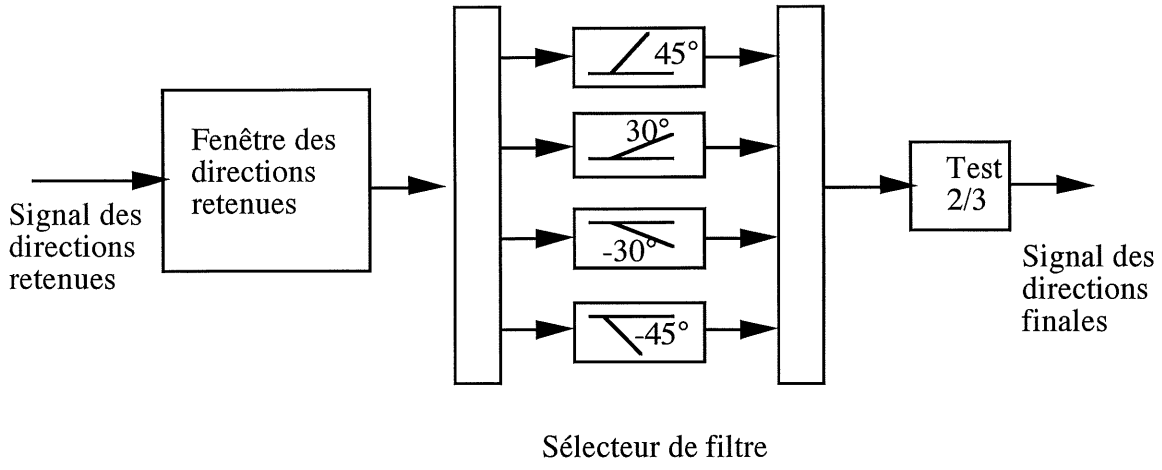


Figure 3.11 Schéma de la partie de fortification

La fenêtre d'image compte trois lignes des directions retenues soit 45^0,30^0,-30^0,-45^0 et deux lignes existantes comme le montre la figure 3.12.

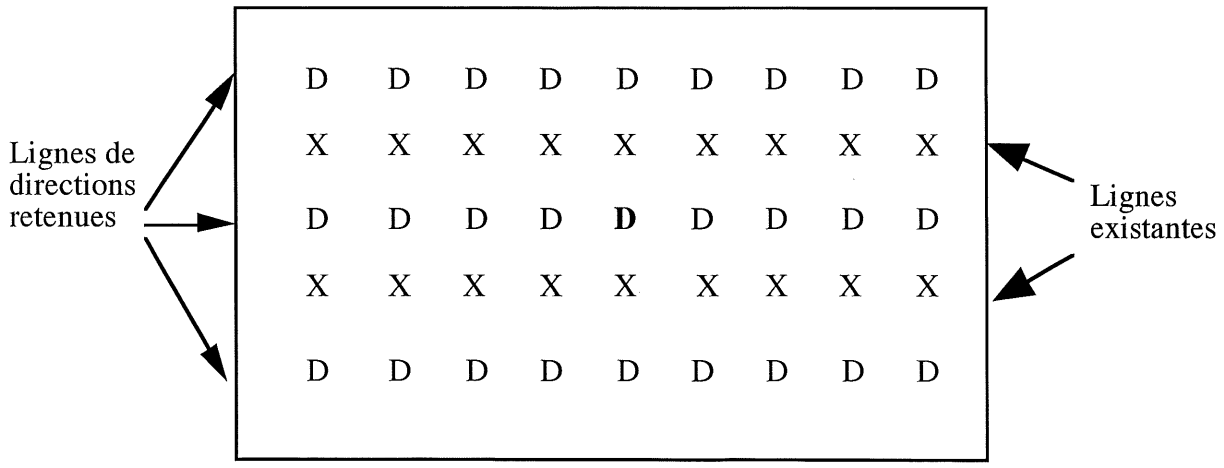


Figure 3.12 La fenêtre des directions retenues

Le sélecteur de filtre envoie les directions retenues à leurs filtres appropriés. Dans cette section, l'on s'en tient seulement aux lignes des directions. Les filtres 2D des directions se définissent comme le montre la figure 3.13

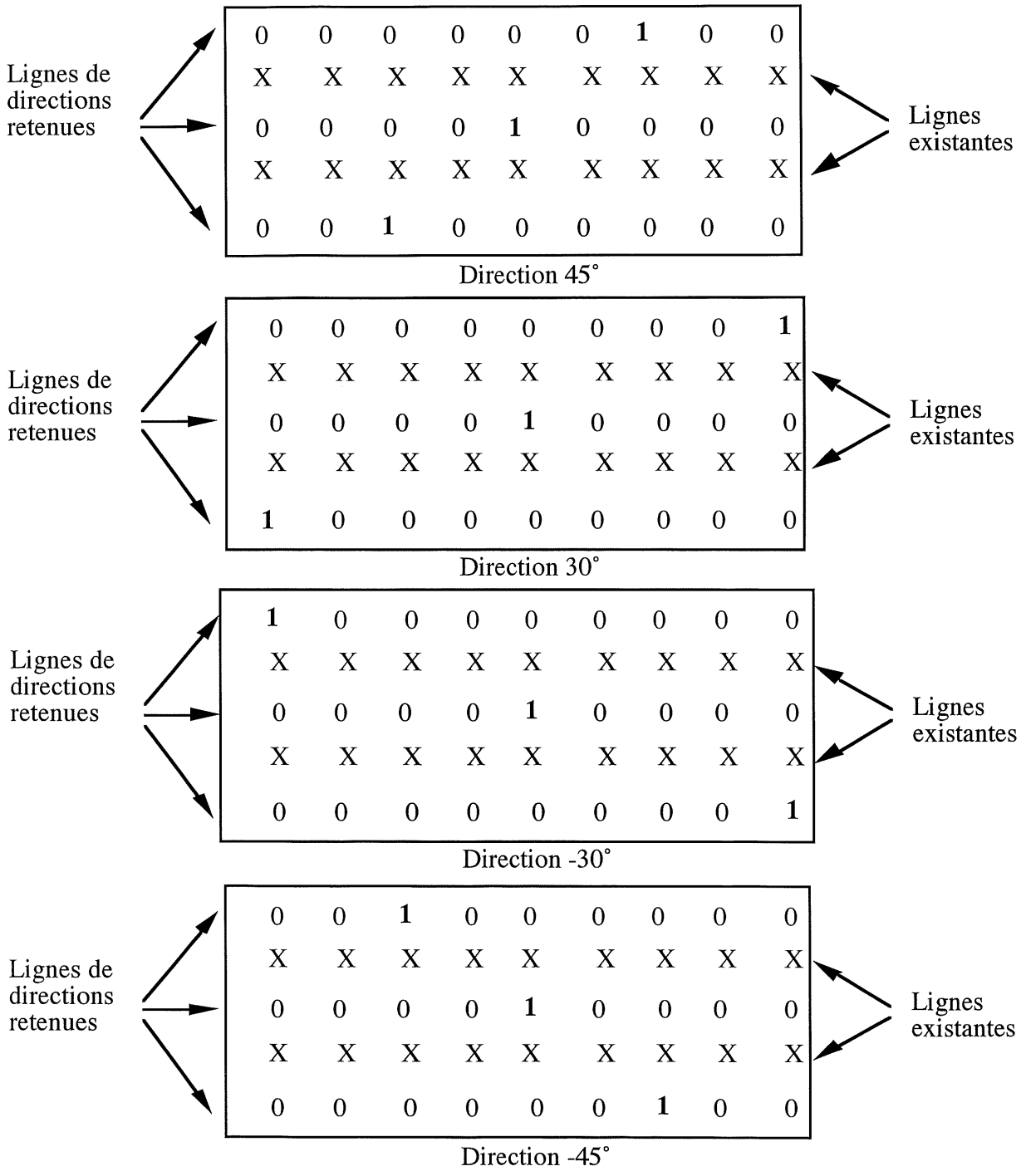


Figure 3.13 Les filtres verticaux-horizontaux selon les directions à consolider

Algébriquement, on a les fonctions de transfert suivantes:

$$h_{45^\circ}(n_h,n_v) = \begin{cases} 1 & pour \quad (n_h,n_v) = (0,0);(2,-2);(-2,2) \\ 0 & ailleurs \end{cases} \tag{3.9}$$

$$h_{30^\circ}(n_h,n_v) = \begin{cases} 1 & pour \quad (n_h,n_v) = (0,0);(4,-2);(-4,2) \\ 0 & ailleurs \end{cases} \tag{3.10}$$

$$h_{-30^\circ}(n_h,n_v) = \begin{cases} 1 & pour \quad (n_h,n_v) = (0,0);(-4,-2);(4,2) \\ 0 & ailleurs \end{cases} \tag{3.11}$$

$$h_{-45^\circ}(n_h,n_v) = \begin{cases} 1 & pour \quad (n_h,n_v) = (0,0);(-2,-2);(2,2) \\ 0 & ailleurs \end{cases} \tag{3.12}$$

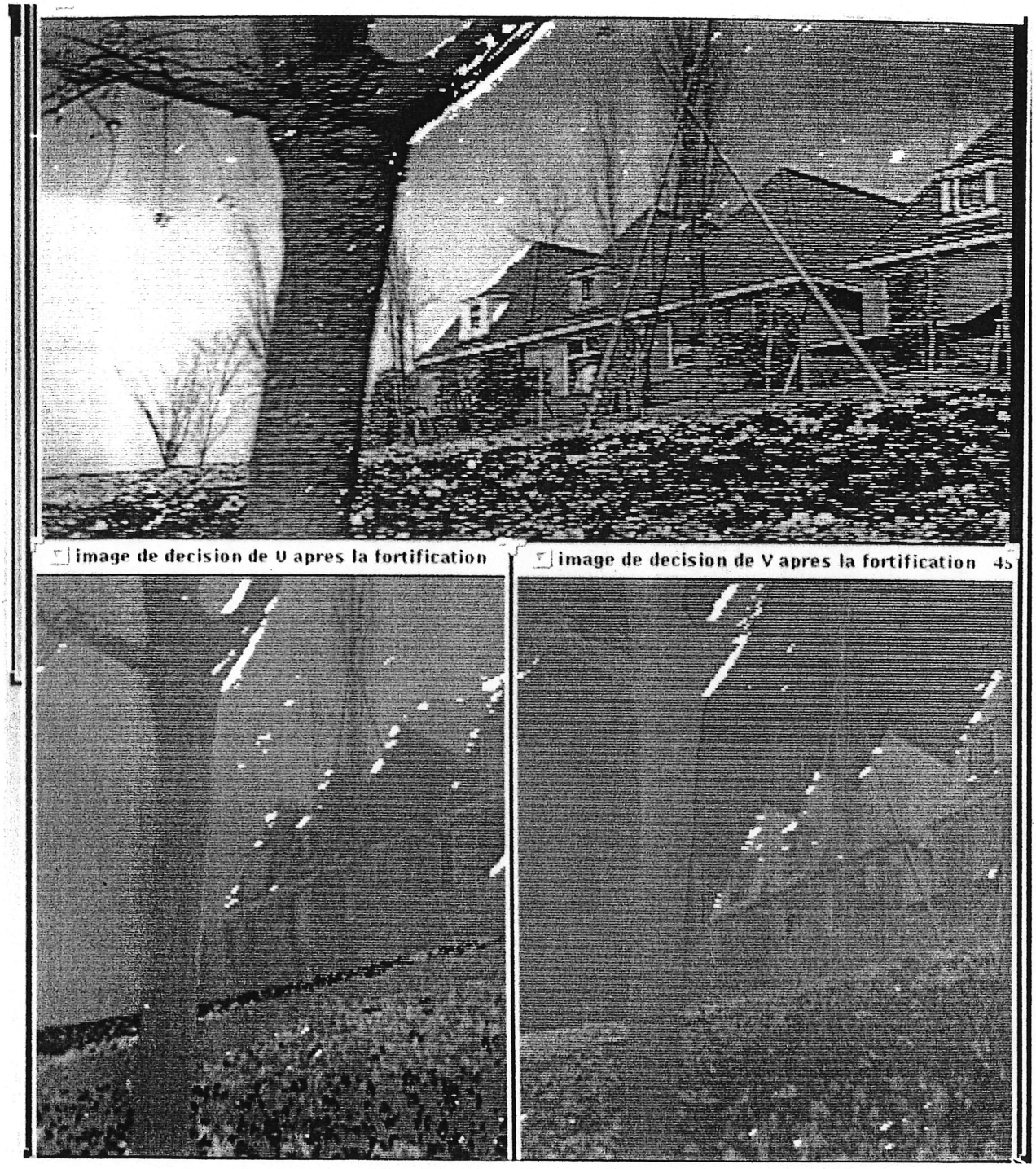
Tous les filtres ont trois coefficients non nuls. Si la réponse à un de ces filtres a plus de deux valeurs non nulles, la direction du filtre en question sera choisie. Si aucune direction n'est choisie, on impose la direction verticale au point considéré. Il arrive parfois que plus d'une direction passe le test 2/3; dans ce cas, on donne la priorité à la direction qui a le plus grand degré (par exemple, dans l'ordre hiérarchique $45^0,30^0,-30^0,-45^0$). Ce processus est non récursif. Il va remplacer et éliminer des directions retenues en même temps. Il est à noter qu'à l'extérieur des contours inclinés, l'interpolation en direction verticale donne le meilleur résultat. La règle d'usage est qu'on donne la priorité à cette direction. Bref, ce processus est représenté par les relations suivantes:

$$S_i = \sum_{n_h=-4}^{n_h=4} \sum_{n_v=-2}^{n_v=2} h_i(n_h,n_v) d_{i;retenu}(n_h,n_v) \quad \text{où i = 45°, 30°, -30°, -45°} \tag{3.13}$$

Si $S_{45^\circ} \geq 2$ alors $D_{final} = D_{45^\circ}$
Sinon et si $S_{30^\circ} \geq 2$ alors $D_{final} = D_{30^\circ}$
Sinon et si $S_{-30^\circ} \geq 2$ alors $D_{final} = D_{-30^\circ}$
Sinon et si $S_{-45^\circ} \geq 2$ alors $D_{final} = D_{-45^\circ}$
Sinon $D_{final} = D_{90^\circ}$ (3.14)

Les figures 3.14, 3.15, 3.16, 3.17 montrent les contours détectés après l'opération de fortification. On voit que la détection est bien localisée sur des contours distincts. Ces contours seront interpolés avec des filtres 3D adaptés à leurs directions.

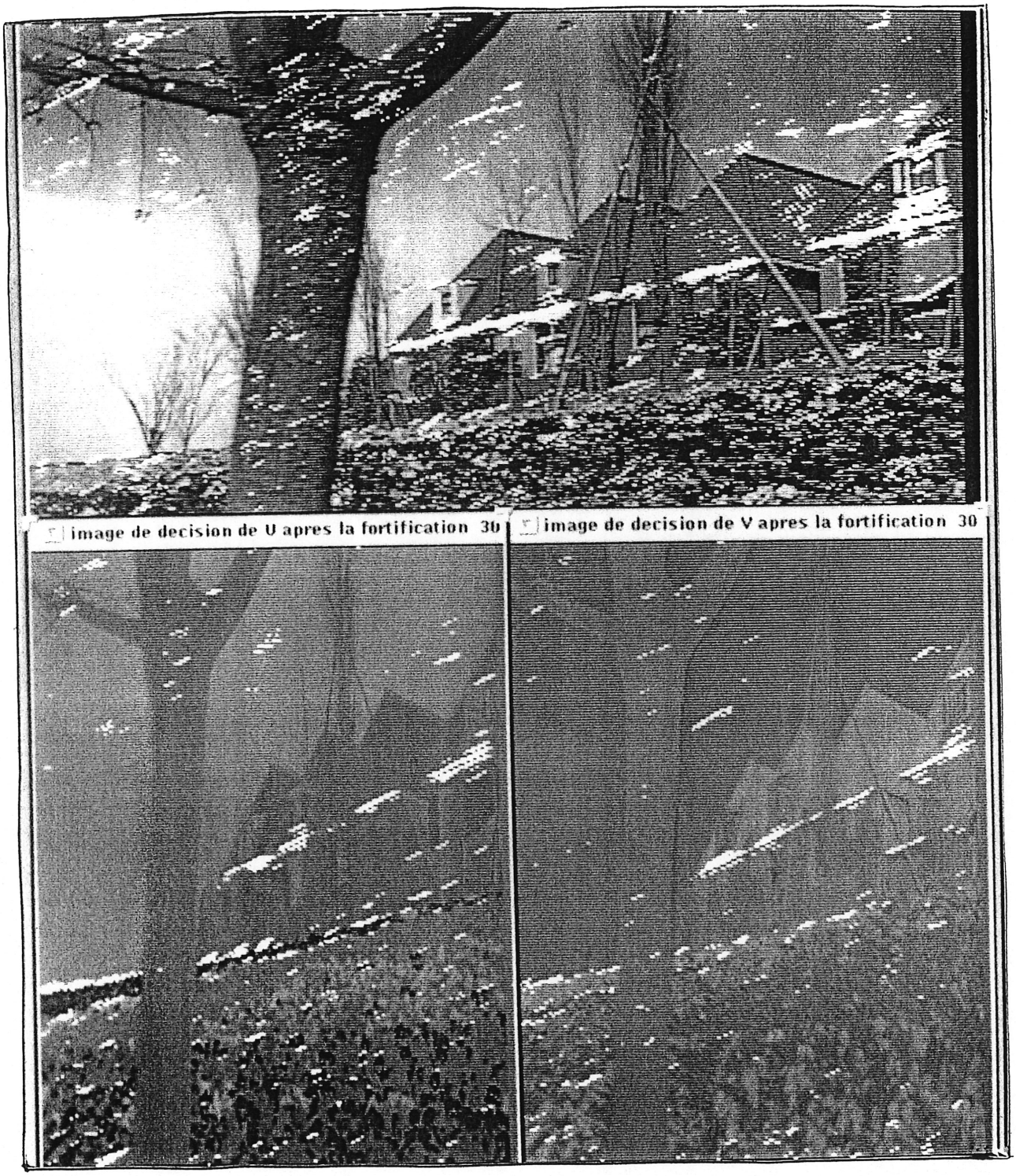
a



b c

Figure 3.14 Image «Flower» avec les contours de 45° détectés après la fortification
a) la composante Y b) la composante U c) la composante V

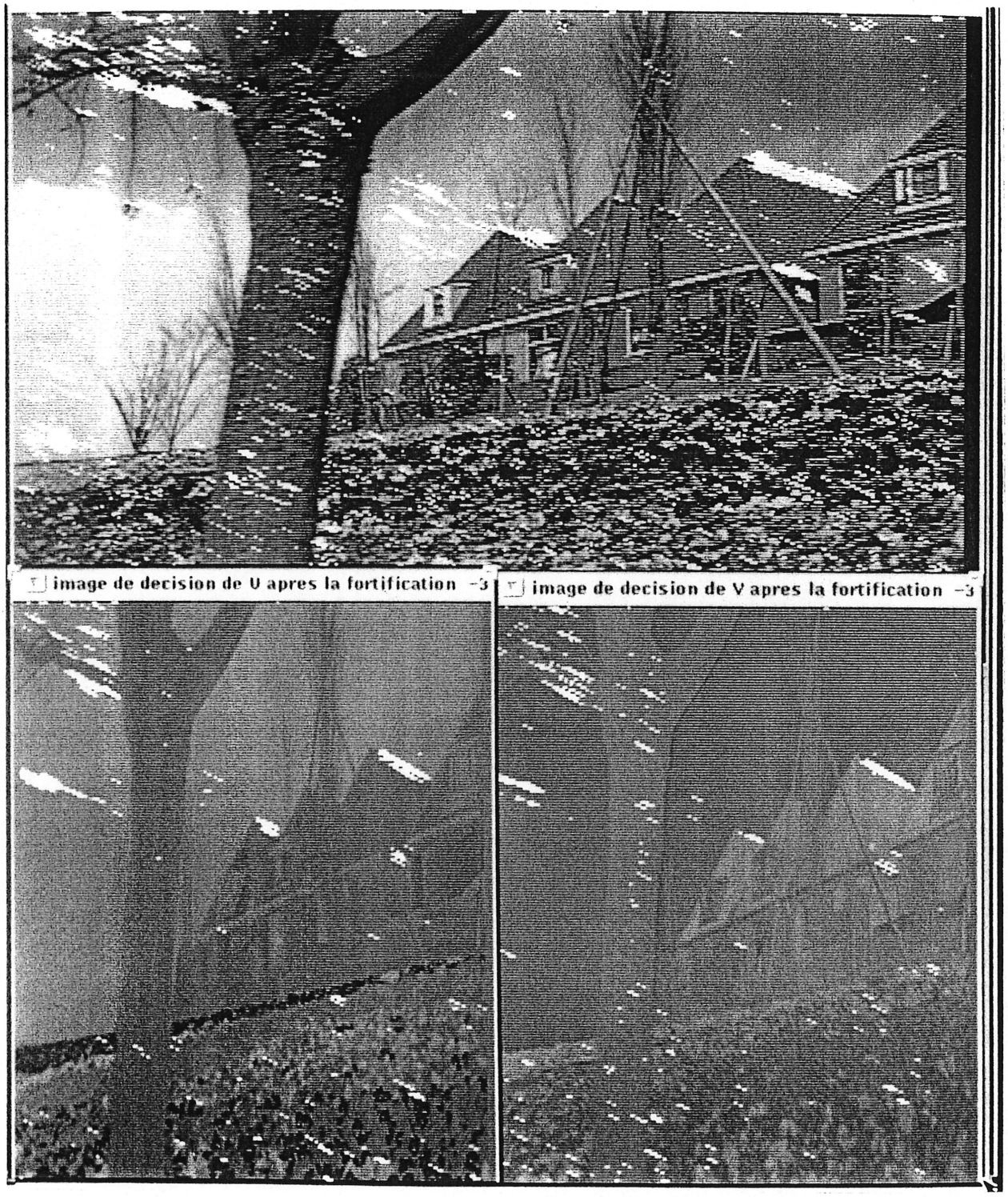
a



b c

Figure 3.15 Image «Flower» avec les contours de 30° détectés après la fortification
a) la composante Y b) la composante U c) la composante V

a



b c

Figure 3.16 Image «Flower» avec les contours de -30° détectés après la fortification
a) la composante Y b) la composante U c) la composante V

a

b c

Figure 3.17 Image «Flower» avec les contours de -45° détectés après la fortification
a) la composante Y b) la composante U c) la composante V

3.2.3 Interpolation selon le contour détecté

Pour interpoler un point donné, on se base sur l'information provenant de trois champs consécutifs et de la direction estimée finale de ce point. Cette direction va déterminer quel filtre il faut utiliser. La figure 3.18 montre le schéma de cette partie.

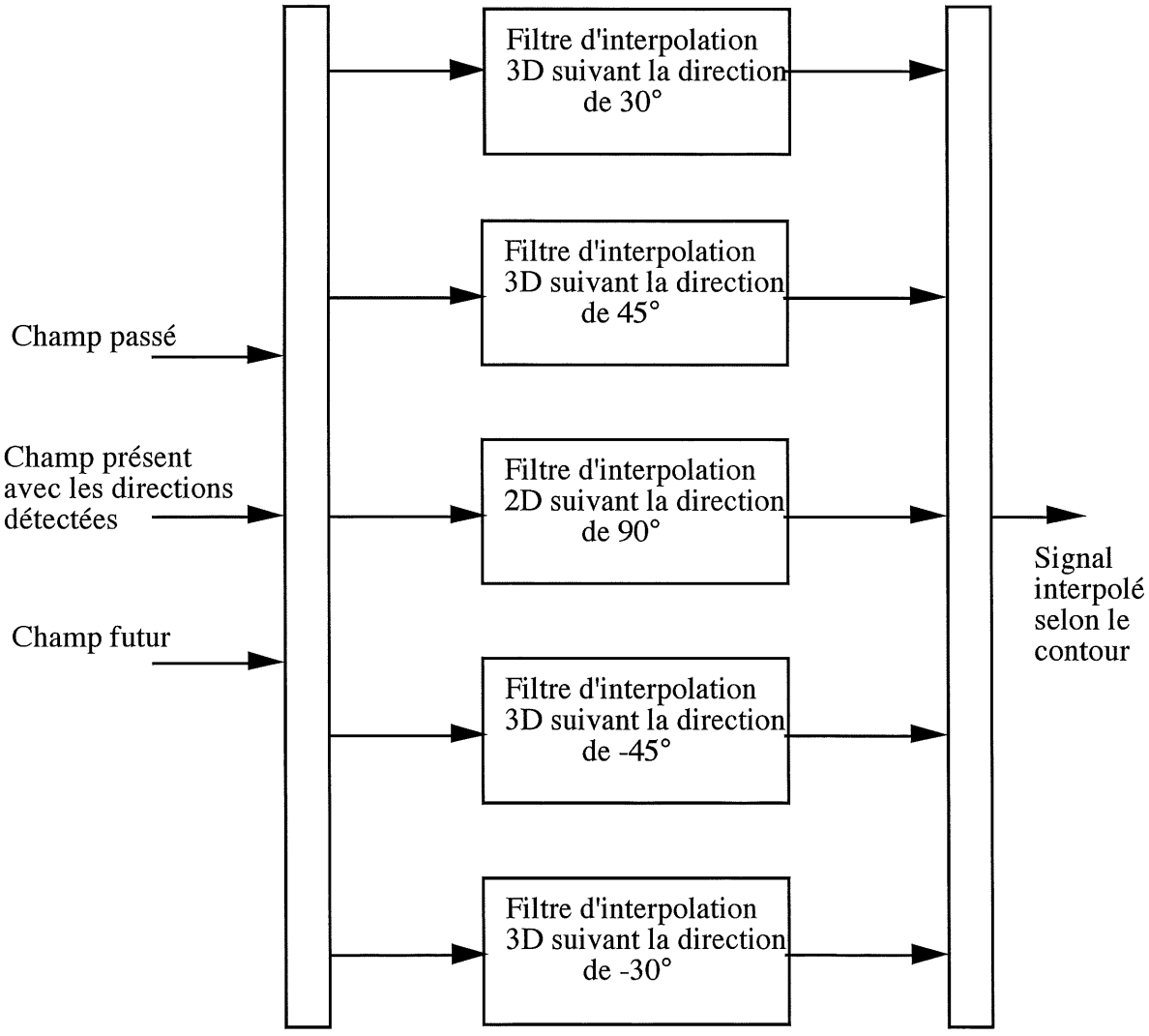


Figure 3.18 Schéma du système d'interpolation selon le contour

3.2.3.1 Filtre à phase linéaire de type FIR

Un filtre $h(n_1,n_2)$ est à phase linéaire (dans ce cas, le filtre est à phase nulle) lorsque sa réponse fréquentielle $H(w_1,w_2)$ est une fonction réelle [21].

$$H(w_1,w_2) = H^*(w_1,w_2) \tag{3.15}$$

Ce type de filtre est fortement utilisé dans le traitement d'image en raison de la sensibilité de l'oeil à la distortion spatiale de phase.

Il est facile de concevoir un filtre à phase linéaire. Selon l'équation (3.15), dans le domaine spatial, on obtient:

$$h(n_1,n_2) = h^*(-n_1,-n_2) \tag{3.16}$$

Étant donné que $H(w_1,w_2)$ est une fonction réelle, cela implique

$$h(n_1,n_2) = h(-n_1,-n_2) \tag{3.17}$$

Le filtre à phase linéaire est donc symétrique à l'origine et, d'après la définition de la transformée de Fourier, $H(w_1,w_2)$ peut s'exprimer comme suit:

$$\begin{aligned} H(w_1,w_2) &= \sum_{n_1=-\infty}^{\infty}\sum_{n_2=-\infty}^{\infty} h(n_1,n_2)e^{-jn_1w_1}e^{-jn_2w_2} \\ &= \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty} 2h(n_1,n_2)\cos(w_1n_1 + w_2n_2) - h(0,0) \end{aligned} \tag{3.18}$$

Comme $h(n_1,n_2) = h(-n_1,n_2) = h(n_1,-n_2) = h(-n_1,-n_2)$, il reste donc seulement un quart des points de $h(n_1,n_2)$ à calculer.

3.2.3.2 Conception des filtres vertico-temporels

L'objectif d'utilisation du filtre V-T est de supprimer les composantes spectrales non désirables dans le domaine fréquentiel. Cet usage équivaut à l'interpolation des lignes manquantes à partir de l'information disponible dans le domaine spatial.

Le signal à l'entrée du filtre est un signal entrelacé suréchantillonné par 2 et le résultat est un signal progressif à la sortie. Le filtre V-T doit notamment éliminer l'effet de recouvrement (aliasing) du signal d'entrée. La figure 3.19 montre le spectre à l'entrée et à la sortie du filtre.

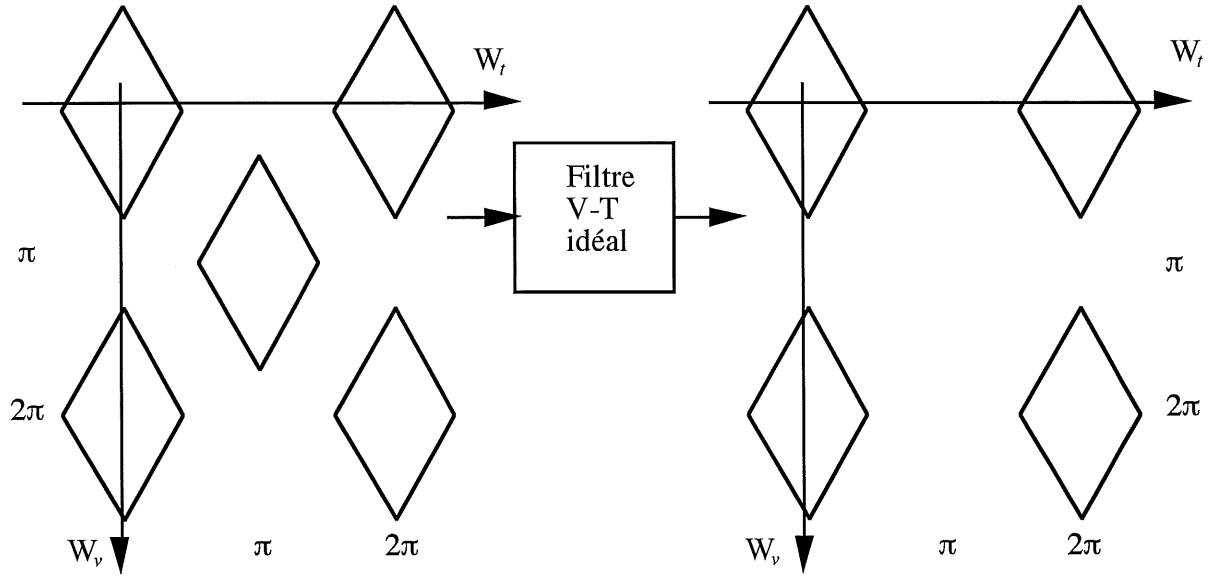


Figure 3.19 Le spectre à l'entrée et à la sortie du filtre V-T idéal

En examinant les figures 3.19, on peut conclure que le filtre V-T doit être un filtre passe-bas vertical et passe-tout temporel. Pour réduire l'effet des traînées de mouvement dues au filtrage temporel, la réponse impulsionnelle de ce filtre dans le domaine du temps doit être équivalente à une impulsion, et ce, au moins pour les composants verticaux à basse fréquence. Cela revient à dire que l'interpolation à basses fréquences verticales est transparente à tout mouvement.

De plus, le filtre V-T doit supprimer les composantes de fréquences situées à (π,π), (π,-π), (-π,π) et (-π,-π) dans le plan W_v, W_t normalisé.

La figure 3.20 donne une allure idéale de ce filtre dans le domaine spectral.

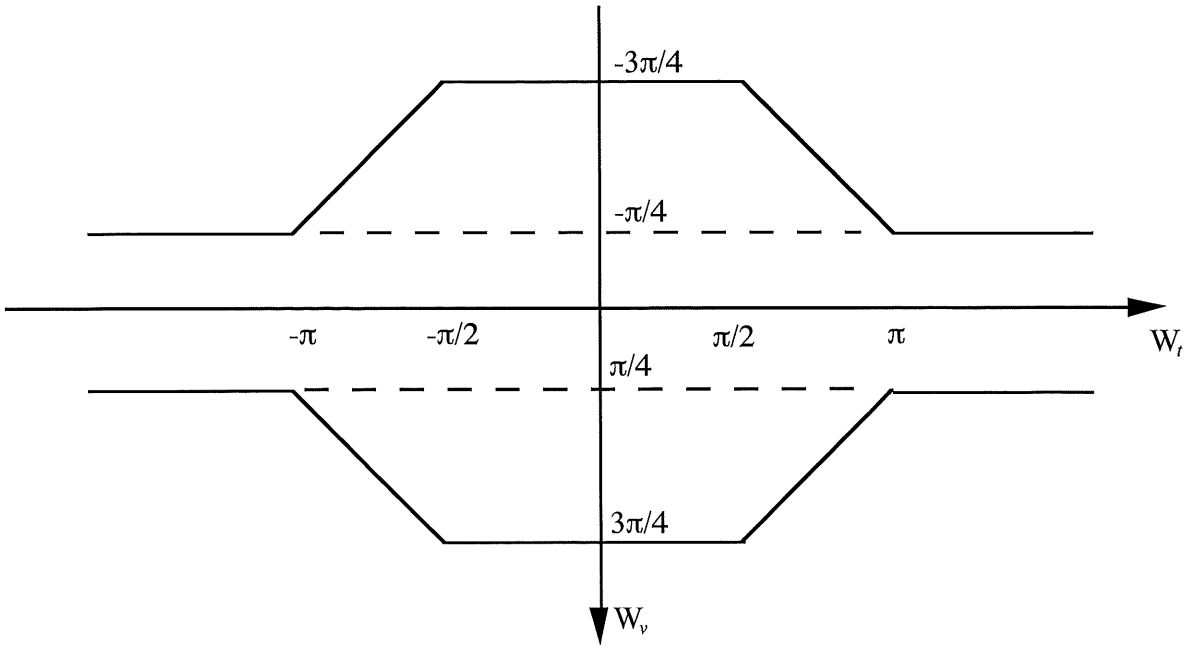


Figure 3.20 La présentation de contour du filtre idéal

On limite le nombre des coefficients du filtre à 4 pour des raisons économiques. Ces coefficients sont présentés sur la figure 3.21.

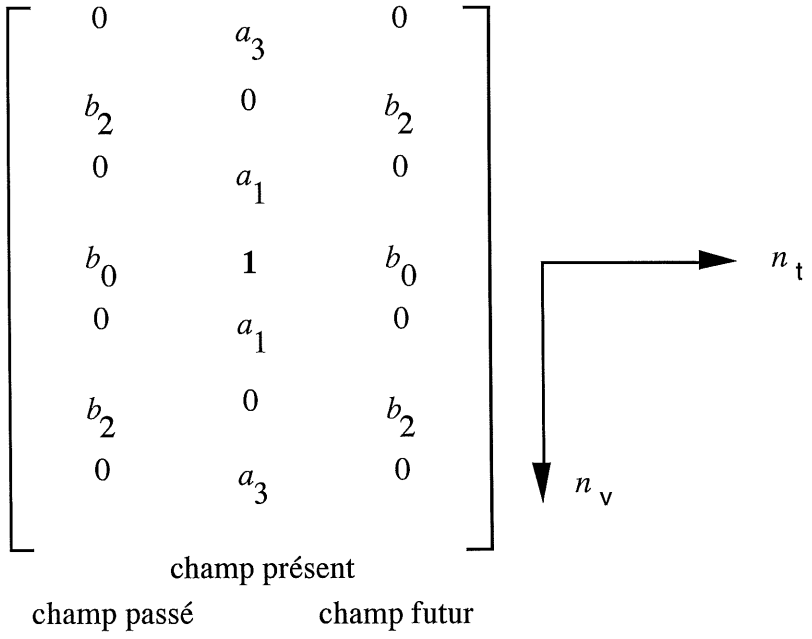


Figure 3.21 La présentation des coefficients du filtre V-T

La valeur du coefficient présente le pourcentage de contribution de niveau de gris du pixel pris par rapport à celui qu'il faut interpoler. Rappelons que plus la ligne contenant des pixels connus est éloignée de la ligne à interpoler, moins son influence est importante.

La fonction de transfert de ce filtre est donnée par l'équation (3.18)

$$H(w_1,w_2) = \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty} 2h(n_1,n_2)\cos(w_1n_1 + w_2n_2) - h(0,0) \qquad (3.18)$$

En remplaçant les coefficients non nuls à chercher et en substituant w_t, w_v pour w_1, w_2 respectivement dans l'équation (3.18), on obtient:

$$H(w_t,w_v) = 1 + 2a_1\cos w_v + 2b_0\cos w_t + 2a_3\cos 3w_v + 4b_2\cos 2w_v\cos w_t \qquad (3.19)$$

En imposant des contraintes appropriées à la fonction de transfert du filtre, on peut trouver ses coefficients. Voici les contraintes qu'on a soigneusement choisies et qui respectent les critères principaux du filtre :

i) Un gain parfait sur l'axe temporel veut dire

$H(w_t,0) = 2$ (car 2 phases)

$1 + 2a_1 + 2a_3 + \cos w_t(2b_o + 4b_2) = 2$

Ce qui implique

$2a_1 + 2a_3 = 1$ (3.20)

$2b_o + 4b_2 = 0$ (3.21)

ii) Un gain nul à (π,π) ne donne pas une nouvelle équation

iii) Développement en série de Taylor dans la direction verticale

Soit $w_t = 0$ et $w_v = x$

On a $\cos w_v = 1 - \frac{x^2}{2} + \cdots$

Substituant dans l'équation (3.19), on obtient

$H(0,x) = a_0 + 2a_1 + 2a_3 + 2b_0 + 4b_2$

$$-x^2(a_1 + 9a_3 + 8b_2)$$

$$+\cdots \qquad (3.22)$$

D'où, on peut tirer une autre équation

$$a_1 + 9a_3 + 8b_2 = 0 \qquad (3.23)$$

iv) L'atténuation -2,5dB à (0,π/2) (le compromis fait à l'aide de Matlab) donne:

$$1 + 2b_0 = 1{,}5 \qquad (3.24)$$

Les équations (3.20), (3.21), (3.23), (3.24) forment un système d'équations linéaires que l'on résoud et qu'on quantifie à un facteur de 128. Les coefficients obtenus sont:

a_1= 56/128

a_3= 8/128

b_0= 32/128

b_2= -16/128

Ce filtre fixe, appelé aussi TSARCRIQ a été antérieurement développé dans le cadre d'un projet conjoint entre Miranda-CRIQ-Agence Spatiale du Canada pour l'interpolation d'image. Ce filtre nous a été communiqué par le Dr. Chon Tam Le Dinh.

On constate que ces coefficients respectent bien le standard de conversion [22]; c'est-à-dire que la somme des contributions dans les deux colonnes externes de $h(n_t, n_v)$ doit être égale à 0. Les images seront floues si on ne respecte pas ce critère.

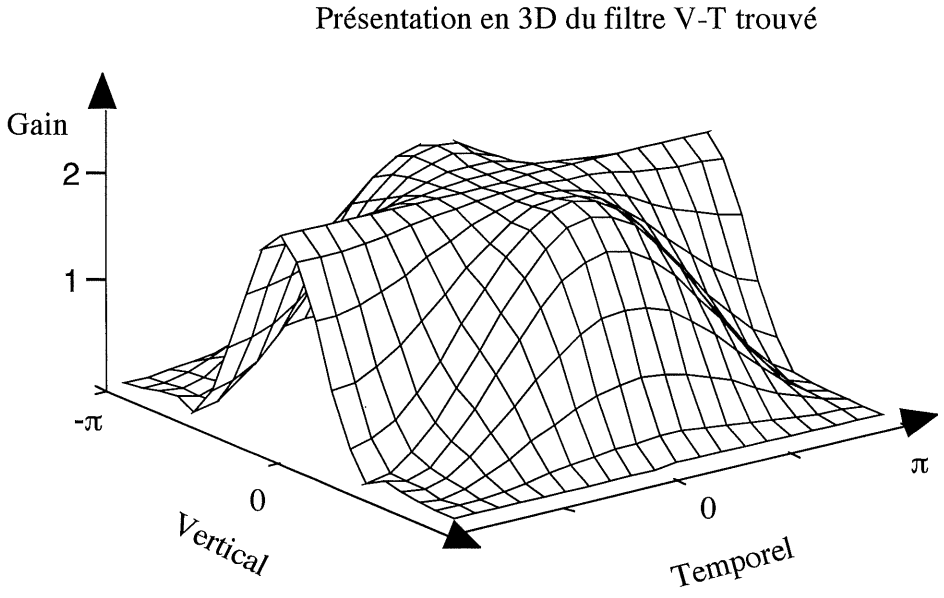


Figure 3.22 La réponse en fréquence du filtre trouvé

Comme on peut le constater (figures 3.22 et 3.23), le filtre V-T coupe bien les fréquences sur l'axe vertical et a un gain parfait sur l'axe temporel. Par contre, il faut signaler que les fréquences diagonales (hautes fréquences verticales et hautes fréquences temporelles) ne sont pas bien filtrées. Cela va introduire l'effet d'escalier sur les régions critiques si le filtre n'est pas adapté à la direction du contour.

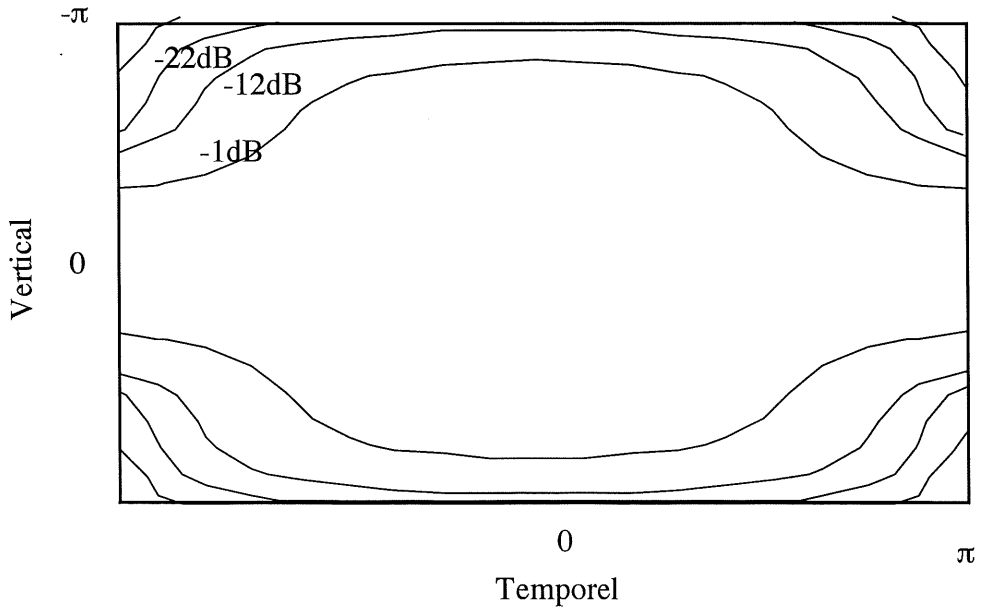


Figure 3.23 Contours isopotentiels V-T

3.2.3.2 Adaptation du filtre 2D selon les directions du contour

Le signal sortant du fortificateur de détection contient les lignes existantes du champ présent qui alternent avec les lignes d'indication des directions du contour. Ces dernières indiquent au sélecteur de filtre de choisir le filtre d'interpolation approprié pour interpoler un point donné.

Vu que les cinq directions nécessitent des filtres d'interpolation adaptés à leur usage, le filtre à quatre coefficients qu'on vient de trouver convient parfaitement bien à la direction verticale (90°). Pour les directions inclinées, on utilise les dérivés du filtre à trois coefficients, car on veut diminuer le nombre de lignes requises. La figure 3.24 présente le filtre simplifié de base en trois champs distincts.

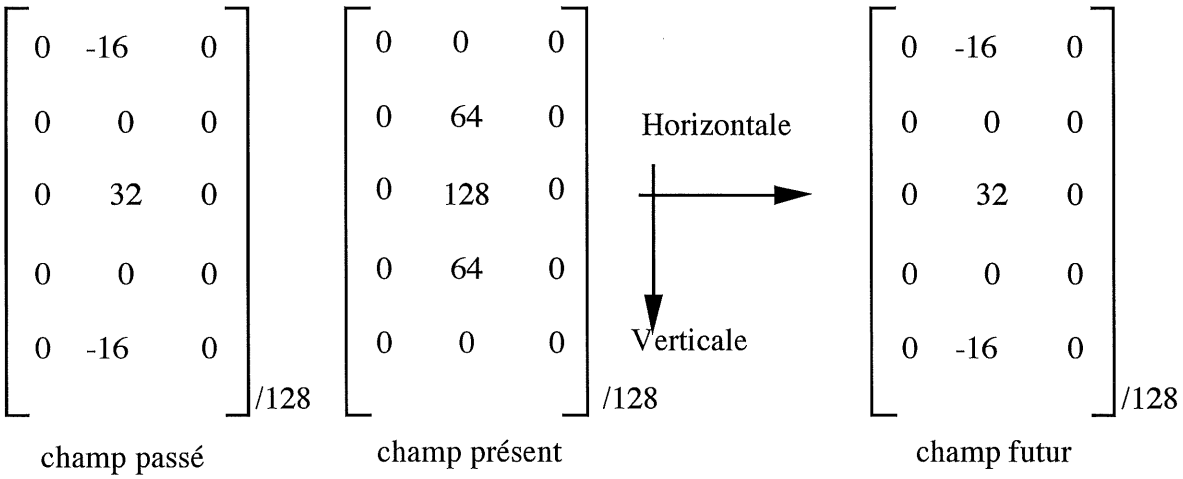


Figure 3.24 Le filtre de base pour les contours inclinés

L'adaptation à la direction estimée du contour se fait en inclinant le filtre de base d'une façon appropriée; et on peut ainsi déduire les configurations du filtre d'interpolation dans les directions $45^0, 30^0, -45^0, -30^0$ telles que le présente la figure 3.25.

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & -16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -16 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} /128 \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 64 & 0 \\ 0 & 0 & 128 & 0 & 0 \\ 0 & 64 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} /128$$

champ passé ou champ futur — champ présent

Filtre d'interpolation pour la direction 45°

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -16 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} /128 \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 64 \\ 0 & 0 & 128 & 0 & 0 \\ 64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} /128$$

champ passé ou champ futur — champ présent

Filtre d'interpolation pour la direction 30°

$$\begin{bmatrix} 0 & 0 & -16 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -16 & 0 & 0 \end{bmatrix} /128 \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 64 & 0 & 0 & 0 \\ 0 & 0 & 128 & 0 & 0 \\ 0 & 0 & 0 & 64 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} /128$$

champ passé ou champ futur — champ présent

Filtre d'interpolation pour la direction -45°

$$\begin{bmatrix} -16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 32 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -16 \end{bmatrix} /128 \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 64 & 0 & 0 & 0 & 0 \\ 0 & 0 & 128 & 0 & 0 \\ 0 & 0 & 0 & 0 & 64 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} /128$$

champ passé ou champ futur — champ présent

Filtre d'interpolation pour la direction -30°

Figure 3.25 Les filtres d'interpolation de quatre directions choisies

Ces filtres ne sont plus V-T, ils sont alors spatio-temporels dans le sens strict.

3.2.4 Détection de mouvement et interpolation temporelle

Dans une image ordinaire, il y a des régions statiques et des régions dynamiques. Le détecteur de mouvement décèle l'état du signal de l'image d'entrée. Il produit un facteur α qui tend vers l'unité si le signal de l'image d'entrée est estimé en mouvement et un facteur α qui tend vers zéro si le signal de l'image d'entrée est fixe. Le facteur α peut être défini comme le degré de mouvement qui détermine la contribution de l'interpolation selon le contour dans le résultat final.

3.2.4.1 Structure et principe du détecteur de mouvement

On a utilisé la valeur absolue de la différence des images successives pour évaluer le degré de mouvement d'une image à l'autre. Le schéma du détecteur de mouvement est présenté à la figure 3.26.

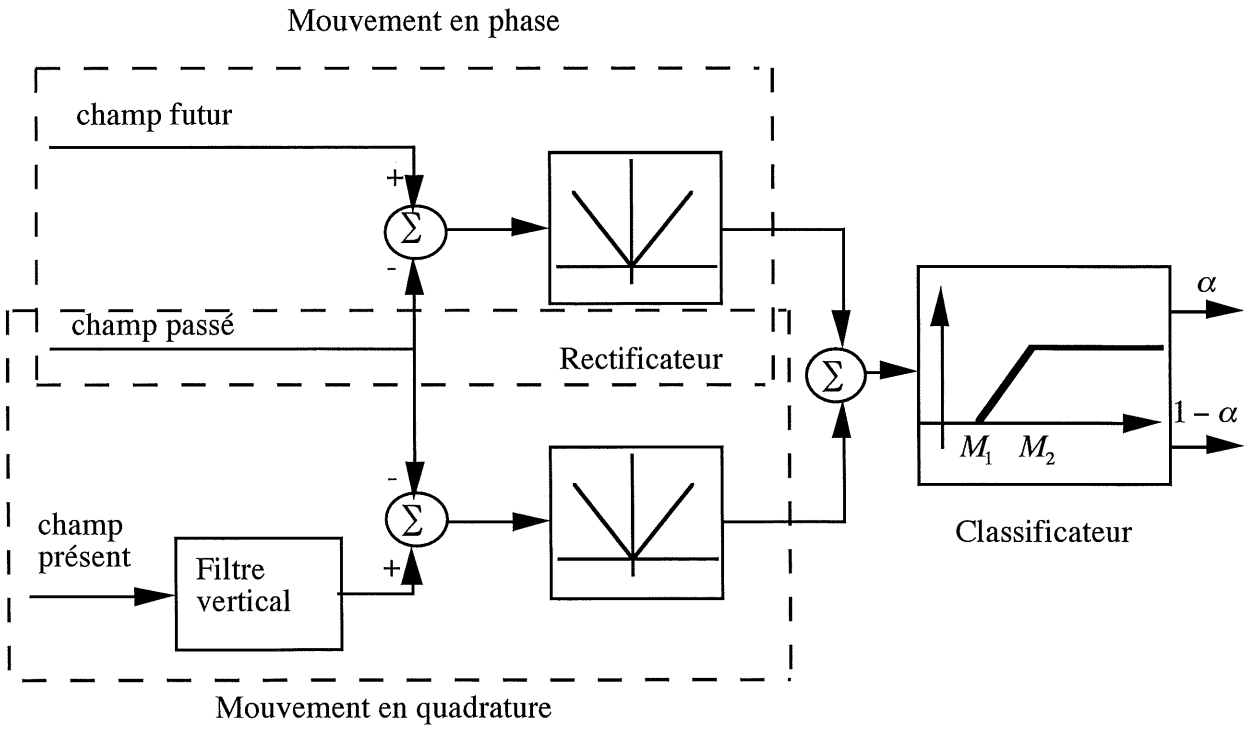


Figure 3.26 Schéma de la partie de la détection de mouvement

Les trois champs successifs du signal entrelacé permettent de calculer le mouvement en phase et en quadrature. La somme de ces deux signaux représente le mouvement total qui entre dans le classificateur. Ce dernier va déterminer le degré de mouvement α variant de 0 à 1 selon deux seuils M_1 et M_2 traités au point 3.2.4.4 dans cette étude.

3.2.4.2 Mouvement en phase

La différence entre le champ passé et le champ futur d'une séquence d'images entrelacées est considérée comme un mouvement en phase [23]. Supposons que l'on veut détecter le mouvement en phase du pixel considéré appartenant au champ présent de la figure 3.27:

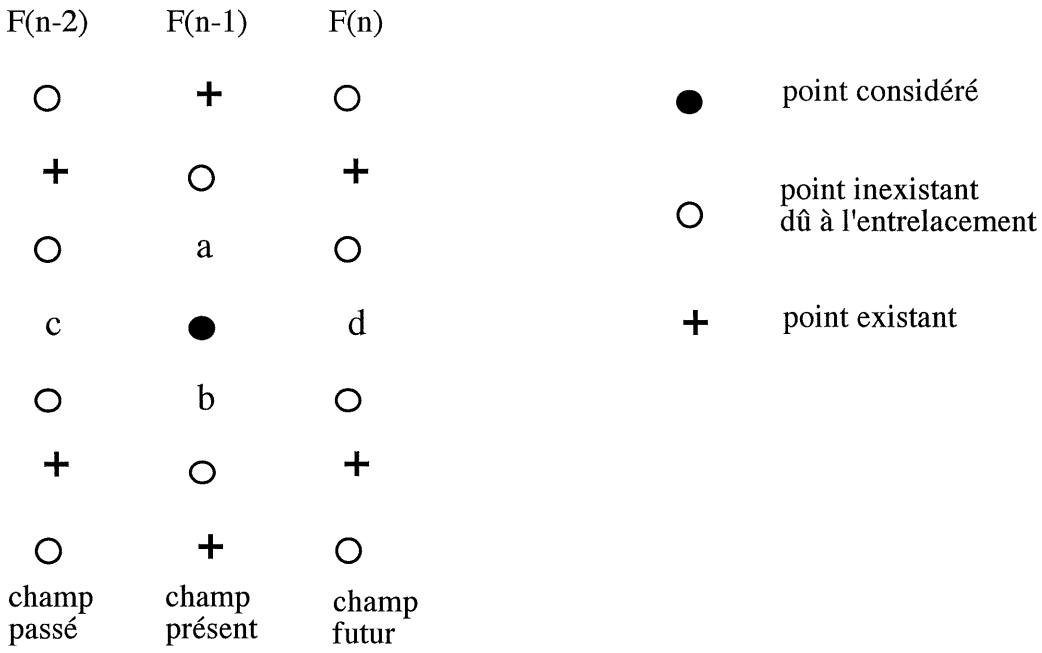


Figure 3.27 Les trois champs d'une séquence d'image entrelacée

Le mouvement en phase de ce pixel est évalué par la valeur absolue de la différence de luminance des pixels "c" et "d" du champs passé et du champs futur respectivement.

$$M_{phase} = |I_c - I_d| \tag{3.25}$$

3.2.4.3 Mouvement en quadrature (ou en 90°)

Tonge [24] a introduit un autre type de mouvement appelé mouvement en quadrature, dont l'intérêt est de tenir compte des mouvements entre deux champs adjacents. Ce mouvement peut facilement échapper à un détecteur de mouvement conventionnel. C'est le cas, par exemple, du mouvement d'une barre qu'on observe souvent lors du défilement des titres électroniques sur écran TV. Considérons une barre verticale qui se déplace horizontalement comme l'illustre la figure 3.28.

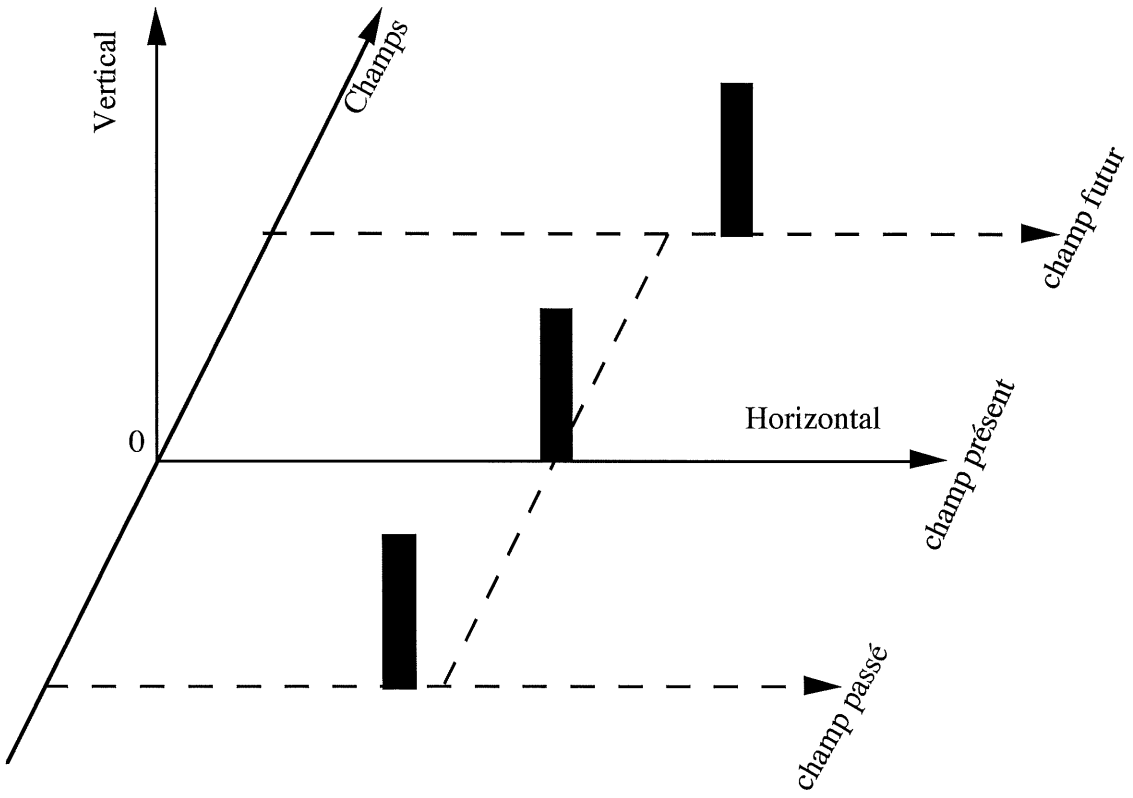


Figure 3.28 Une barre verticale se déplace horizontalement à travers trois champs

On voit qu'il n'y a pas de différence de luminance entre le champ passé et le champ futur du point considéré. Pourtant, la barre est en mouvement. Elle passe à travers le champ présent. Ceci montre l'absense de détection de deux mouvements, un mouvement entre le champ passé et le champ présent et l'autre entre le champ présent est le champ futur. Il est donc important de décomposer le détecteur de mouvement en phase et en quadrature.

Comme la valeur de luminance du point considéré est inexistante, il faut d'abord l'interpoler par un filtre vertical d'ordre 1. Puis on doit calculer la valeur absolue de différence entre la luminance du point considéré et celle du champ passé. À l'aide de la figure 3.27, on a:

$$M_{quadrature} = \left| \frac{I_a + I_b}{2} - I_c \right| \tag{3.26}$$

3.2.4.4 Classificateur de mouvement

Le classificateur est une fonction non linéaire qui détermine le facteur α à partir du mouvement total calculé M_T. La figure 3.29 montre cette relation.

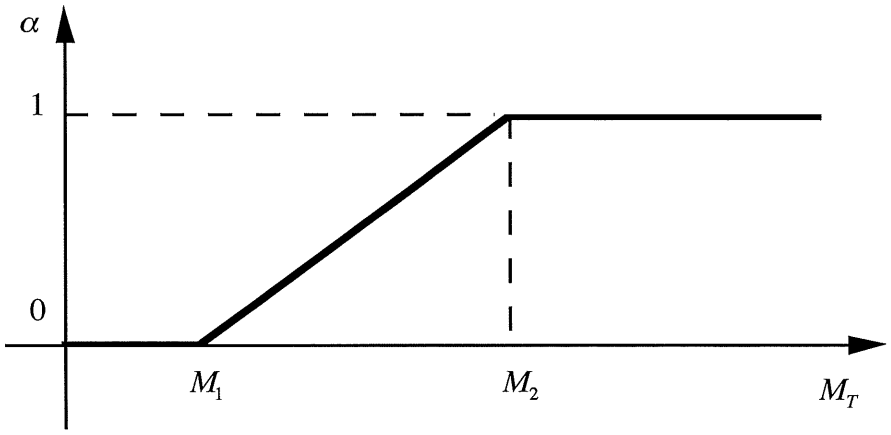


Figure 3.29 La fonction non linéaire

Le classificateur classe le mouvement total calculé M_T en trois régions:

i) La première région $M_T < M_1$: Il n'y a pas ou très peu de mouvements. Le classificateur produit un facteur α qui tend vers zéro.

ii) La deuxième région $M_1 < M_T < M_2$: Il y a un faible mouvement. Dans ce cas, le facteur de mouvement α suit une fonction linéaire tel que:

$$\alpha = \frac{M_T - M_1}{M_2 - M_1} \tag{3.27}$$

où M_1 et M_2 sont des seuils convenables que l'on a soigneusement choisis à l'aide de l'histogramme de l'image en mouvement.

iii) La troisième région $M_2 < M_T$: Il y a un fort mouvement. Quand le mouvement dépasse le seuil M_2, le classificateur impose le facteur α à 1.

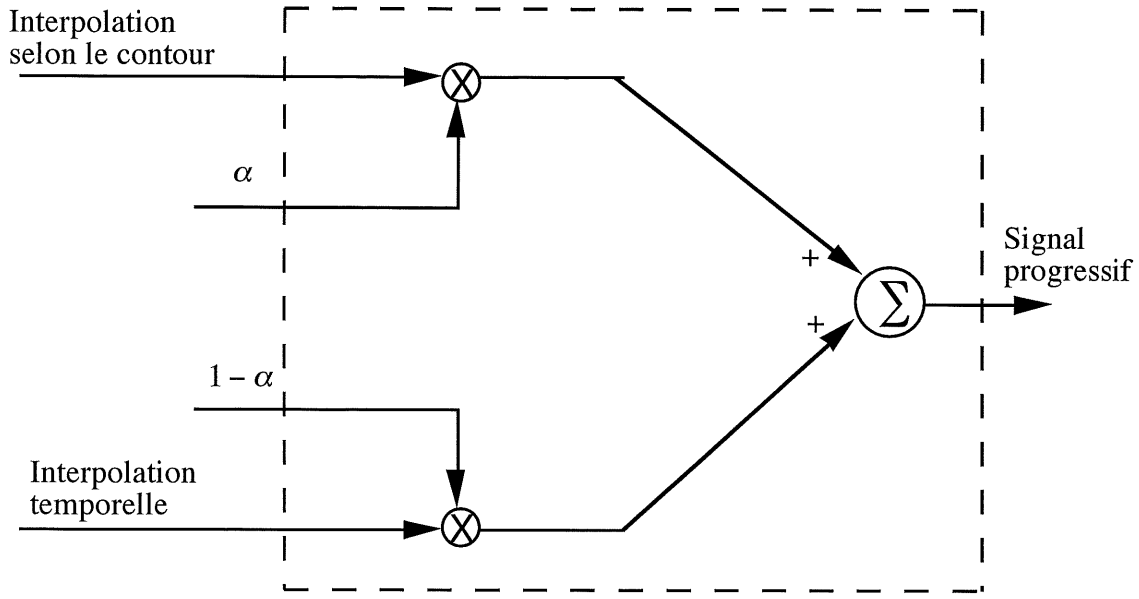
3.2.4.5 Interpolation temporelle

Tel que souligné précédemment, le filtre temporel donne une meilleure interpolation pour une image statique. Quand il n'y a pas de mouvement, la moyenne de luminance du champ passé et du champ futur donne exactement la luminance du champ présent. Théoriquement, il n'y a pas de différence entre le champ passé et le champ futur; on aurait pu remplacer directement la valeur du champ passé dans le champ présent. En pratique, on doit prendre la moyenne pour atténuer les erreurs dues aux bruits. D'après la figure 3.27, l'interpolation temporelle du point considéré se calcule comme suit:

$$I_{point\_considéré} = (I_c + I_d) / 2 \qquad (3.28)$$

3.2.5 Adaptation spatio-temporelle

L'adaptation entre l'interpolation selon le contour et l'interpolation temporelle se fait à l'aide de deux opérateurs arithmétiques simples (multiplication et sommation) comme le montre la figure 3.30.



Adaptation spatiotemporelle

Figure 3.30 Adaptation entre deux interpolations

Le détecteur de mouvement fournit les facteurs α et $1-\alpha$ qui déterminent la contribution de l'interpolation selon le contour et la contribution de l'interpolation temporelle dans le résultat final.

Cette relation peut s'exprimer mathématiquement comme suit:

$$I_{po\,int\_considéré} = \alpha . I_{\substack{int\,erpolation\_selon \\ le\_contour}} + (1-\alpha) . I_{\substack{int\,erpolation \\ temporelle}} \qquad (3.29)$$

3.3 Conclusion

Les techniques que l'on a utilisées dans la conception de doubleur de lignes sont plutôt visuelles et intuitives qu'analytiques. La pratique permet à choisir des filtres appropriés. Les résultats de la simulation du système proposé sont présentés au chapitre 5. On a vu qu'il existe beaucoup de théories sur l'interpolation basées sur de belles équations mathématiques; qui toutefois ne peuvent être appliquées ici ou elles marchent mal.

Le présent doubleur de lignes est limité à cinq directions. Par conséquent, il ignore certains contours. Pour les contours dont la direction est proche de 90^0, les erreurs de détection n'affectent pas beaucoup l'interpolation. Par contre, pour les contours dont la direction est proche de zéro, les erreurs changent énormément les valeurs interpolées.

Dans la partie de fortification de la détection, on aurait dû définir des filtres d'élimination et de remplissage de façon indépendante, ce qui rendrait plus efficace chaque opération de filtrage. Par ailleurs, on pourrait élargir la fenêtre des directions retenues pour aussi tenir compte de pixels plus éloignés.

Tel que vu précédemment, chaque bloc du système est simple et fonctionne de façon indépendante. On a effectué des tests et des essais sur chaque bloc. En général, chacun donne de bons résultats comme illustrent les figures 3.14 - 3.17. Si on accepte de prendre un certain risque avec le raisonnement que l'oeil n'est pas sensible à la direction temporelle, on peut utiliser seulement deux champs au lieu de trois, F(n) étant le champ présent et F(n-1) étant le champ passé et champ futur.

4

CONVERSION À HAUTE RÉSOLUTION

La conversion en haute résolution d'un signal entrelacé peut se faire par une approche directe ou indirecte. Une approche directe possible est basée sur un filtre linéaire vertical-temporel à polyphase travaillant sur le signal entrelacé. L'approche indirecte, plutôt basée sur le doubleur de lignes qui lui procure un signal progressif, interpole par filtrage linéaire séparable en haute résolution. Cette deuxième approche est fortement recommandée car elle donne un meilleur résultat. En effet, l'interpolation est non linéaire et elle est rendue adaptative par le doubleur de lignes.

Dans le présent travail, on aborde les deux approches, d'abord la méthode indirecte et ensuite la méthode directe.

4.1 Conversion à TVHD d'un signal progressif (méthode indirecte)

Comme l'image à l'entrée du convertisseur est une image progressive, on n'a besoin que d'interpolation spatiale, c'est-à-dire d'insérer des zéros entre deux échantillons existants et de faire du filtrage passe-bas linéaire pour supprimer des composantes à hautes fréquences. Les filtres 1D de Kaiser constituent les éléments de base de la conversion faite ici, comme le montre la figure 4.1.

Ici, le signal progressif de 525 lignes est converti au standard nord-américain de TVHD; soit 787,5 lignes, soit 1050 lignes. Cependant, dans le cas présent, on a choisi le format de 1050 lignes de 1920 pixels par image. En fait au moment de la présente recherche, les standards TVHD n'étaient pas encore bien définis.

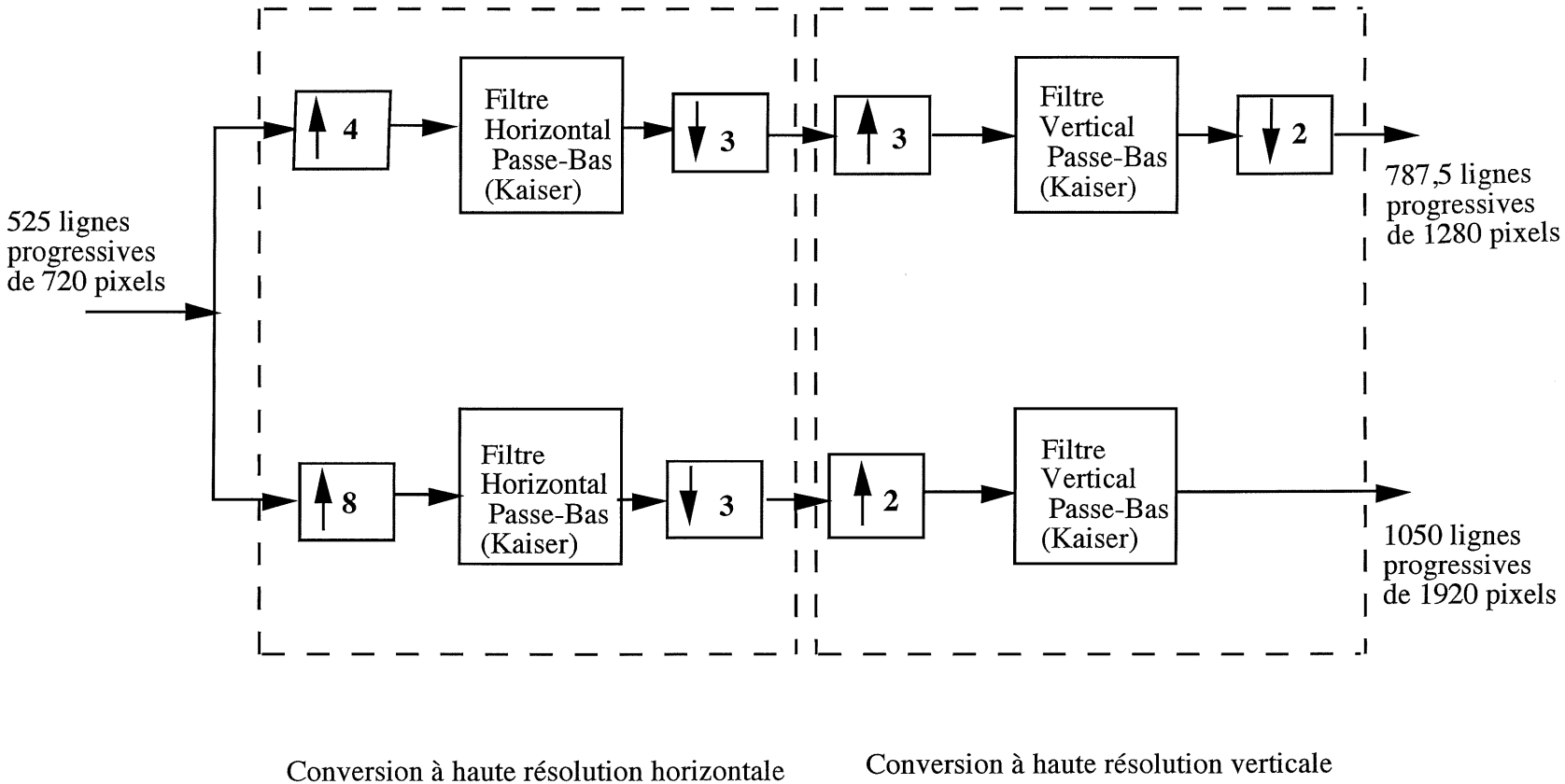


Fig. 4.1 Schéma des systèms de conversion à haute résolution d'un signal progressif

Le format D-1 de la recommandation 601-1 du CCIR spécifie que la largeur de bande des signaux 625/50 ou 525/60 est approximativement de 6 MHz pour la luminance et de 3 MHz pour les signaux de chrominance [25]. La conception des filtres nécessaires illustrés par la figure 4.1 doit tenir compte de ces spécifications.

Il est à noter que la qualité d'image TVHD obtenue dépend fortement de la performance du doubleur de lignes utilisé.

4.1.1 Interpolation horizontale

L'interpolation horizontale est basée principalement sur le filtre passe-bas qui doit respecter des critères spécifiques de CCIR 601 tels que présentés sur la figure 4.2.

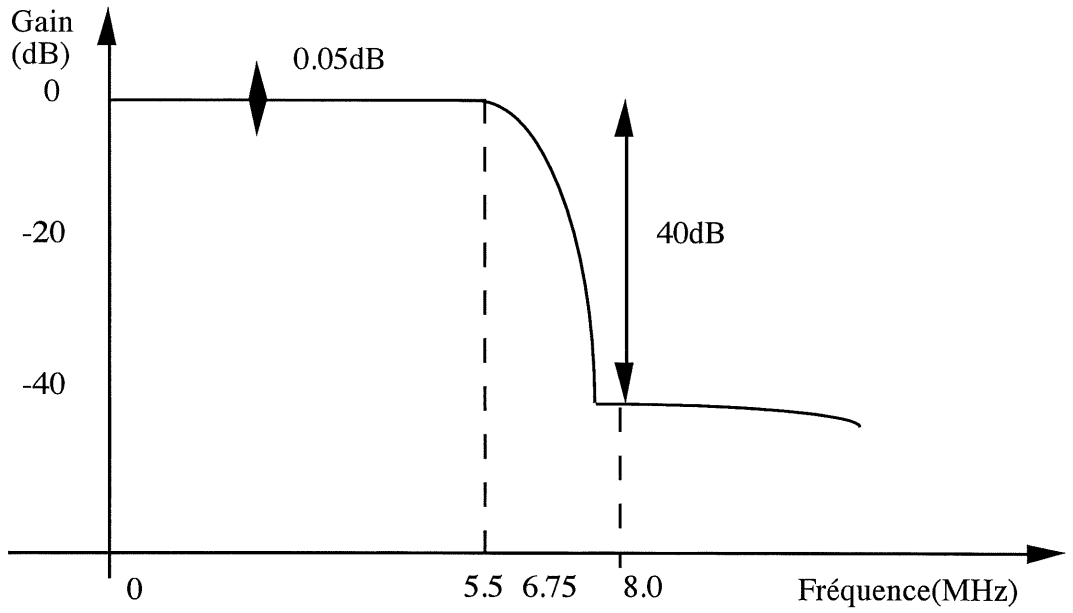


Figure 4.2 Les critères de la réponse en fréquence du filtre 1D horizontal

Numériquement, on a:

α_p = 0,05 dB; α_a = 40 dB

fbande = 6,75 MHz

fp = 5,50 MHz

fa = 8,00 MHz

Comme un signal suréchantillonné d'ordre 8 se trouve à l'entrée du filtre horizontal, ce dernier doit comporter de huit phases séparables impliquant beaucoup de coefficients. Ce filtre peut être conçu facilement par la fenêtre de Kaiser [26]. Sa réponse en fréquence simulée est illustrée dans la figure 4.3 et les calculs impliqués se trouvent en annexe B.

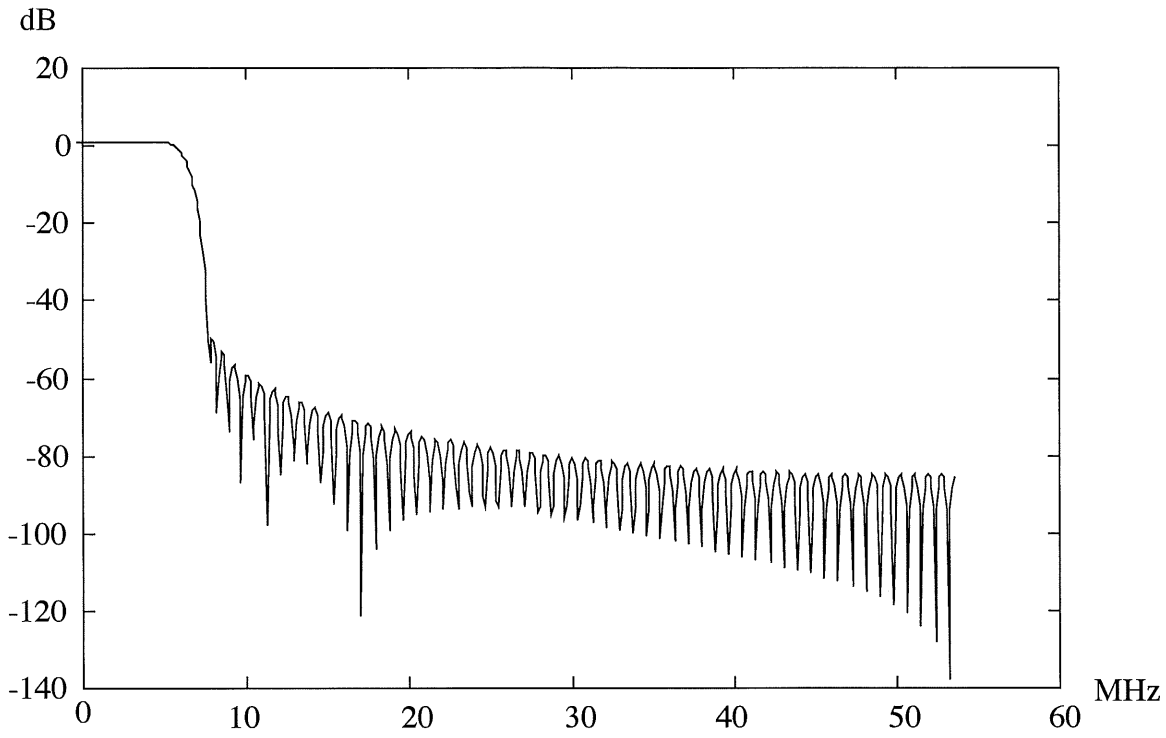


Figure 4.3 La réponse en fréquence du filtre horizontal trouvé

4.1.2 Interpolation verticale

Il n'y a pas de critère spécifique pour le filtre vertical. Comme l'interpolation verticale est moins exigeante que l'interpolation horizontale, on peut utiliser les mêmes critères que ceux du filtre horizontal et on obtient des résultats plus que satisfaisants.

Ce filtre vertical ne comporte que de deux phases et nécessite peu de coefficients par rapport au filtre horizontal. Ses coefficients quantifiés à un facteur de 128 et transposés sur l'axe horizontal sont présentés au tableau 4.1.

$\frac{1}{128}$	-3	0	5	0	-8	0	13	0	-25	0	81	128	81	0	-25	0	13	0	-8	0	5	0	-3

Tableau 4.1 Les coefficients du filtre vertical d'ordre supérieur

Ce type de filtre conçu similairement à celui vu au chapitre 2, est maintenant appliqué à un signal progressif où il a produit de très bons résultats.

4.2 Conversion en TVHD d'un signal entrelacé (interpolation directe)

Au début de notre étude, on a tenté la conversion directe bidirectionnelle (V-T). L'objectif poursuivi était de convertir le signal de 525 lignes entrelacées directement au signal 787,5 lignes progressives (TVHD). Des recherches ont été entreprises pour trouver un filtre bidimensionnel à polyphase. Le schéma bloc d'un tel système est montré à la figure 4.6.

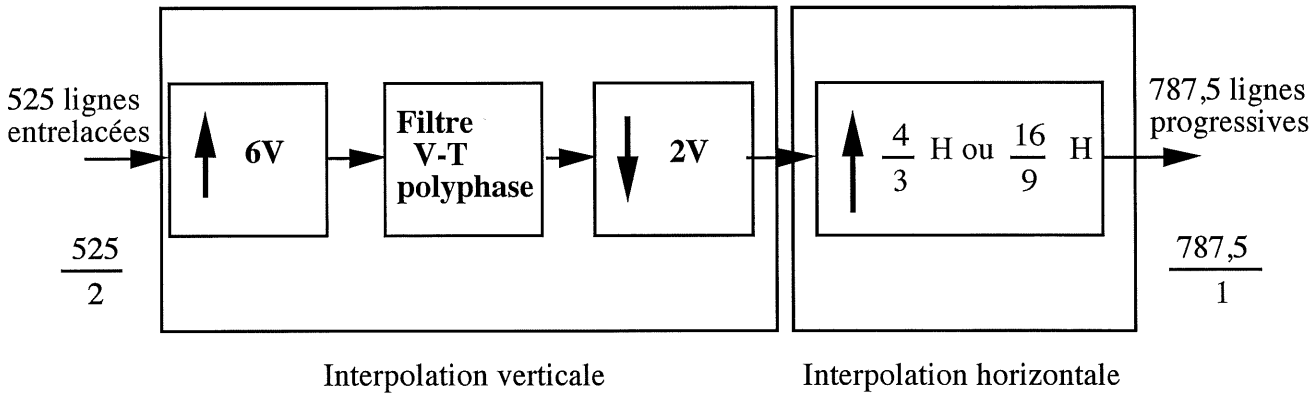


Figure 4.4 Le système d'interpolation directe

Normalement, l'interpolation horizontale se fait sans problème. Toute l'information nécessaire se trouve dans le même champ; donc avec un filtre unidimensionnel de Kaiser on peut très bien traiter le signal. Ce filtre est identique à celui qu'on a vu précédemment.

Dans le cas de l'interpolation verticale, on a besoin d'interpoler des lignes manquantes se trouvant dans des champs différents, ce qui fait intervenir un filtre vertical-temporel. Ce filtre bidimensionnel n'a pas de spécification précise à respecter sauf qu'il doit minimiser le brouillage

des mouvements. De plus, il est très limité à cause des contraintes de coût: on se limite à un filtre de 3 champs.

Puisque l'interpolateur fait intervenir un suréchantillonneur, un filtre bidimensionnel à phase linéaire de type FIR et un sous-échantillonneur, on va rappeler brièvement les définitions de ces concepts pour ensuite élaborer sur la conception du filtre V-T.

2.2.1 Suréchantillonnage et sous-échantillonnage

Dans le domaine temporel [27], le suréchantillonnage d'ordre 6 d'un signal x(n) consiste à insérer cinq échantillons de valeur zéro entre tous les échantillons originaux successifs de x(n) (Tableau 4.2a). Dans le domaine fréquentiel, la présentation spectrale du signal à la sortie du suréchantillonneur est un déploiement du spectre du signal d'entrée dont la période est réduite d'un facteur 6 par rapport à celle du signal d'entrée (Tableau 4.2b).

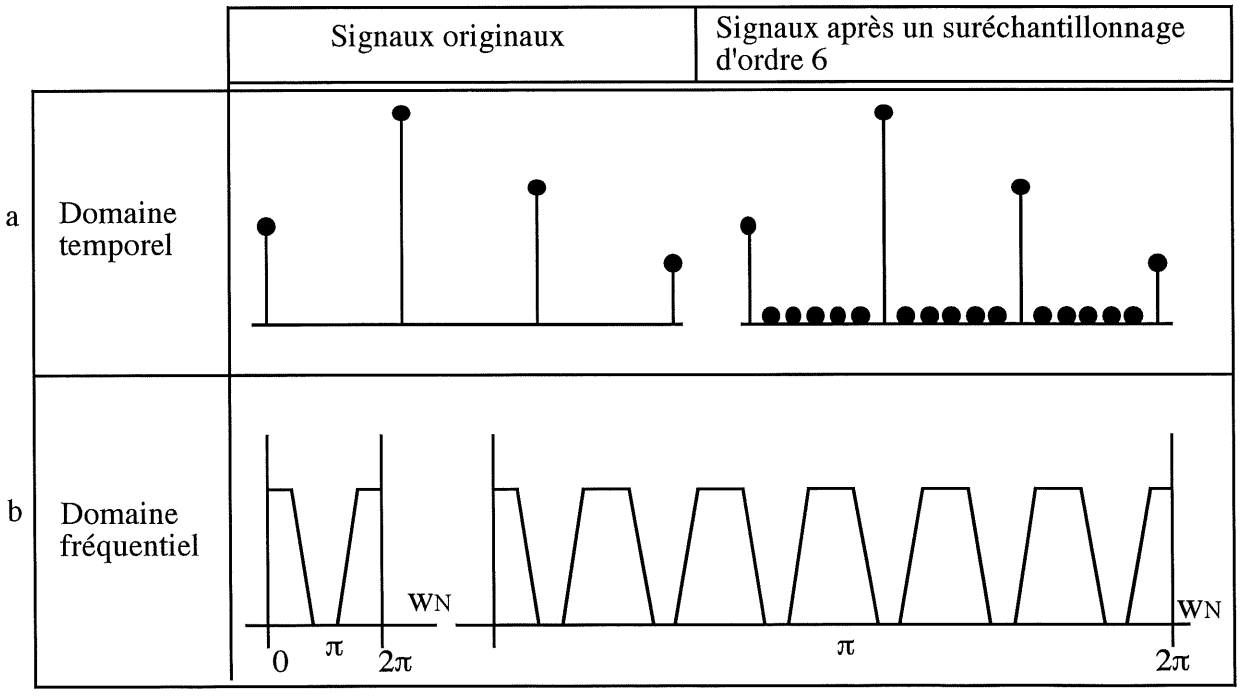


Tableau 4.2 Exemple d'un suréchantillonnage d'ordre 6

Le sous-échantillonnage est un processus inverse de celui du suréchantillonnage. Dans ce cas, on procède à la suppression d'échantillons au lieu de l'insertion d'échantillons dans le domaine temporel et la période est augmentée plutôt que réduite. Il est à noter que ce faisant, il y a perte d'information. Les tableaux 4.3a et 4.3b montrent un sous-échantillonnage d'ordre 2 dans les domaines temporel et fréquentiel.

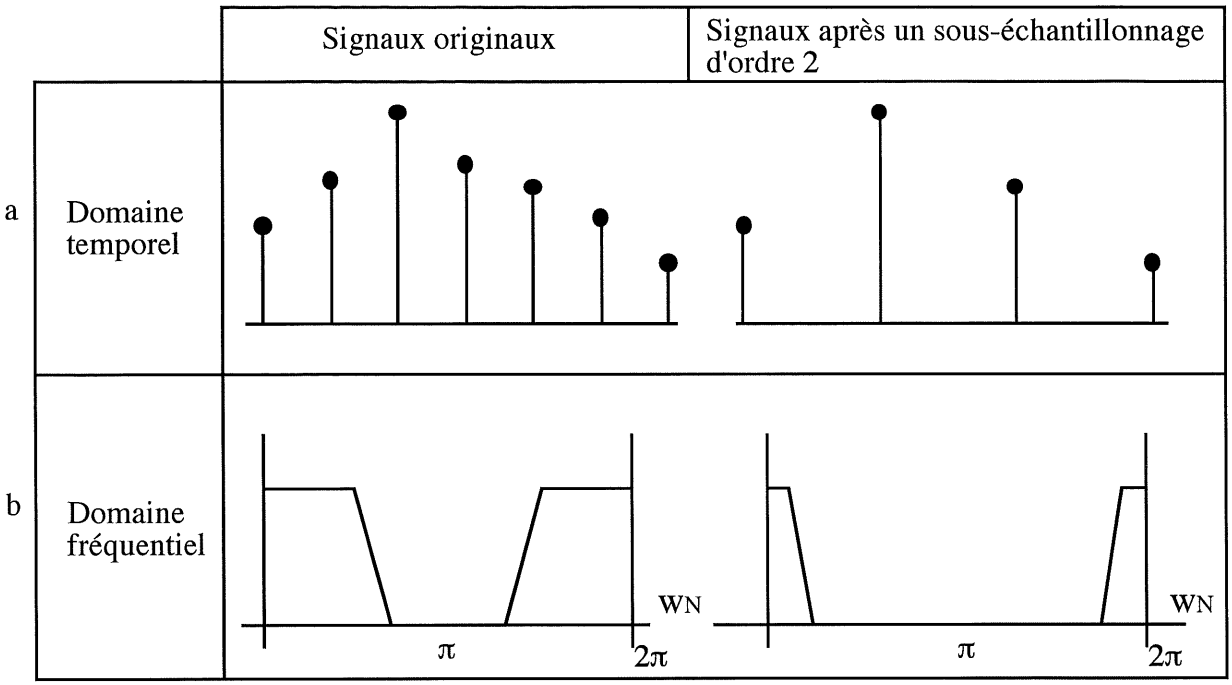


Tableau 4.3 Exemple d'un sous-échantillonnage d'ordre 2

2.2.3 Conception du filtre vertical-temporel

La conception du filtre V-T polyphase est similaire à celle qu'on a vue au chapitre 3. Sauf qu'elle est beaucoup plus complexe et qu'elle exige davantage de compromis. Ici, il ne s'agit plus d'une ligne à interpoler mais de cinq à la fois. Donc, ce filtre compte 6 phases distinctes. La figure 4.7 montre le spectre vertical-temporel normalisé à l'entrée et à la sortie du filtre.

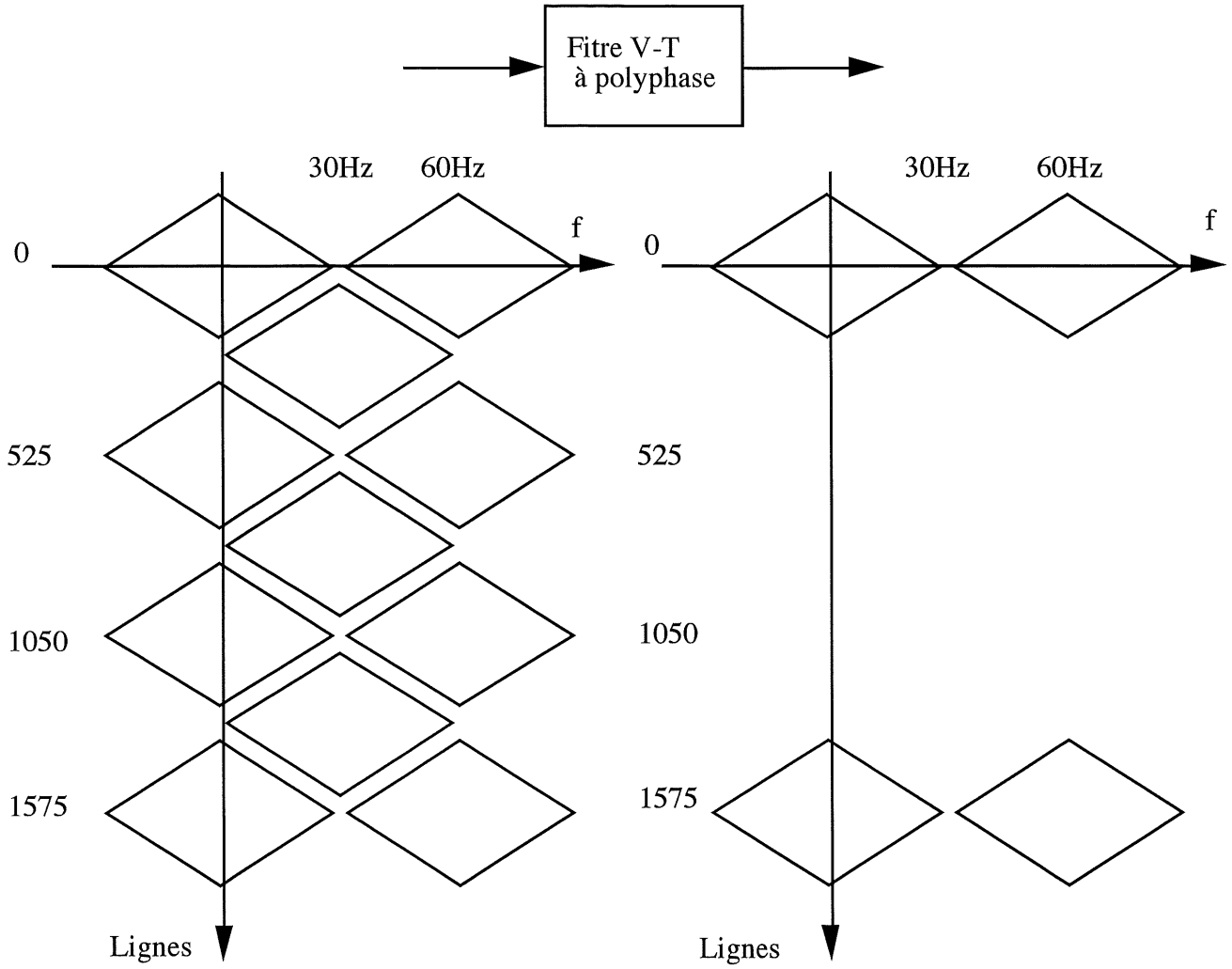


Figure 4.5 Le spectre V-T à l'entrée et sortie du filtre idéal

En examinant la figure 4.5, on peut conclure que ce filtre V-T n'est autre qu'un filtre passe-bas vertical et passe-tout temporel. Ce filtre idéal doit réussir à enlever des composantes spectrales non désirables qui causent souvent de l'interférence.

Dans le plan temporel-vertical, les coefficients du filtre sont montrés à la figure 4.6. Ici, le nombre des coefficients est limité arbitrairement à 21. La valeur du coefficient présente le pourcentage de contribution de niveau de gris du pixel pris par rapport à celui à interpoler. Rappelons que plus la ligne contenant des pixels connus est proche de la ligne à interpoler plus son influence est importante sur cette dernière.

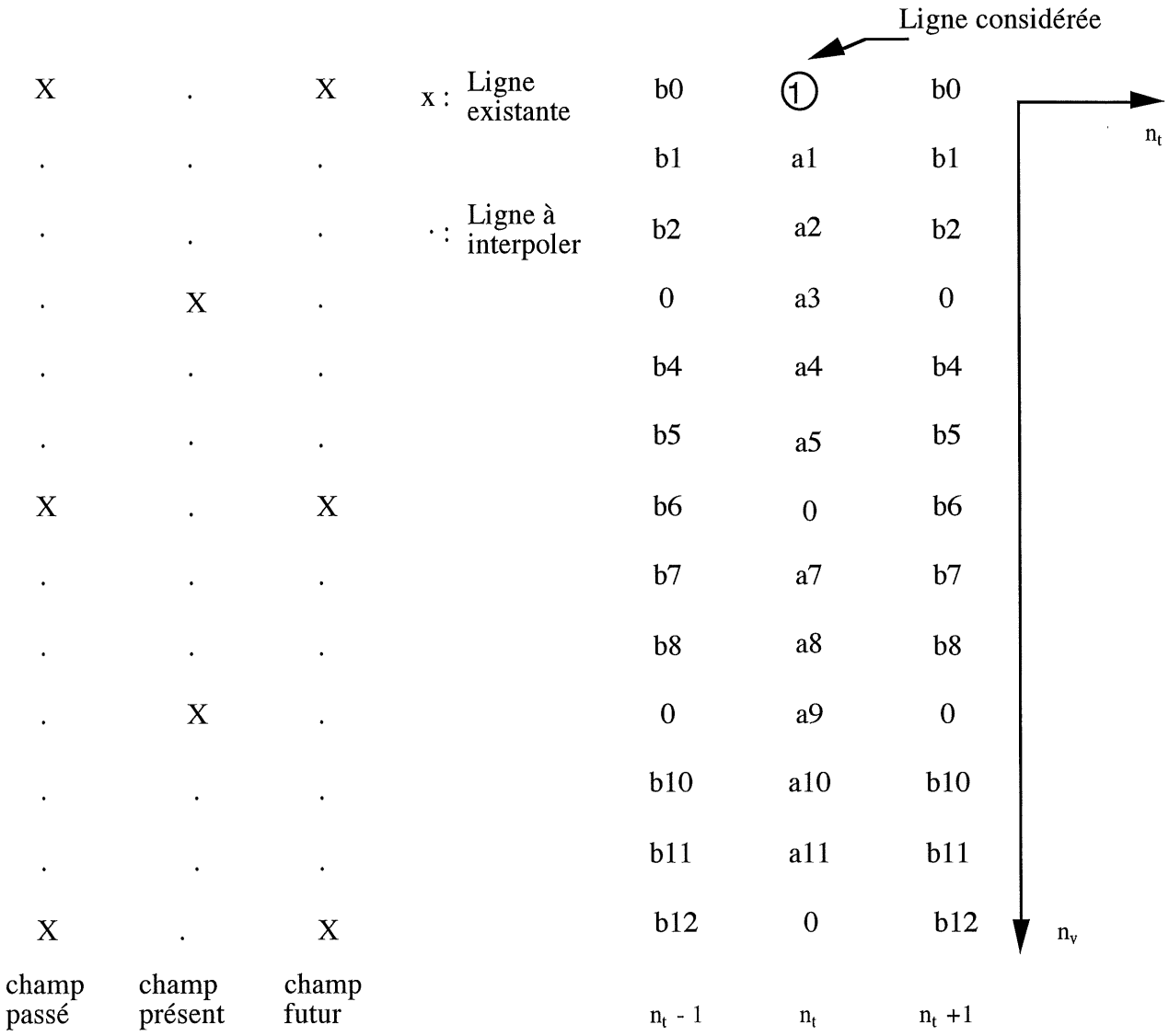


Figure 4.6 La présentation des coefficients du filtre V-T en question

La fonction de transfert de ce filtre est donnée par l'équation (3.18)

$$H(w_1,w_2) = \sum_{n_1=0}^{\infty}\sum_{n_2=0}^{\infty} 2h(n_1,n_2)\cos(w_1n_1 + w_2n_2) - h(0,0) \qquad (3.18)$$

En remplaçant les coefficients non nuls à chercher et en substituant w_1,w_2 par w_t,w_v dans l'équation (3.18), on a:

$$H(w_t,w_v) = 1 + 2\{a_1\cos w_v + a_2\cos 2w_v + a_3\cos 3w_v + a_4\cos 4w_v$$
$$+ a_5\cos 5w_v + a_7\cos 7w_v + a_8\cos 8w_v + a_9\cos 9w_v$$

$$+ a_{10}\cos 10w_v + a_{11}\cos 11w_v\}$$
$$+ 2b_0\cos w_t$$
$$+ 2\{\ b_1\cos(w_t + w_v) + b_2\cos(w_t + 2w_v) + b_4\cos(w_t + 4w_v)$$
$$+ b_5\cos(w_t + 5w_v) + b_6\cos(w_t + 6w_v) + b_7\cos(w_t + 7w_v)$$
$$+ b_8\cos(w_t + 8w_v) + b_{10}\cos(w_t + 10w_v)$$
$$+ b_{11}\cos(w_t + 11w_v) + b_{12}\cos(w_t + 12w_v)\ \} \qquad (4.1)$$

En utilisant l'identité trigonométrique appropriée, on obtient:

$$\mathrm{H}(w_t, w_v) = 1 + 2\{a_1\cos w_v + a_2\cos 2w_v + a_3\cos 3w_v + a_4\cos 4w_v$$
$$+ a_5\cos 5w_v + a_7\cos 7w_v + a_8\cos 8w_v + a_9\cos 9w_v$$
$$+ a_{10}\cos 10w_v + a_{11}\cos 11w_v\}$$
$$+ 2b_0\cos w_t$$
$$+ 2\cos w_t\{\ b_1\cos w_v + b_2\cos 2w_v + b_4\cos 4w_v + b_5\cos 5w_v$$
$$+ b_6\cos 6w_v + b_7\cos 7w_v + b_8\cos 8w_v + b_{10}\cos 10w_v$$
$$+ b_{11}\cos 11w_v + b_{12}\cos 12w_v\ \} \qquad (4.2)$$

La conception du filtre V-T se fait dans le domaine spectral. En imposant des contraintes de façon appropriée dans sa réponse fréquentielle (éq. 4.2), on obtient un système d'équations linéaires à résoudre.

Par exemple, en imposant

$H(w_t, 0) = 6$, pour avoir un gain parfait sur l'axe temporel

on peut en tirer 2 équations:

$$a_1 + a_2 + a_3 + a_4 + a_5 + a_7 + a_8 + a_9 + a_{10} + a_{11} = 2{,}5 \qquad (4.3)$$

et

$$\frac{1}{2}b_0 + b_1 + b_2 + b_4 + b_5 + b_6 + b_7 + b_8 + b_{10} + b_{11} + b_{12} = 0 \qquad (4.4)$$

Voici les contraintes que l'on a soigneusement choisies pour $\mathrm{H}(w_t, w_v)$

$$H(w_t, \pi) = 0 \qquad (4.5)$$

$$H(w_t, \pi/3) = 0 \qquad (4.6)$$

$$H(w_t, \pi/2) = 0 \tag{4.7}$$

$$H(w_t, 5\pi/6) = 0 \tag{4.8}$$

$$H(w_t, 5\pi/12) = 0 \tag{4.9}$$

$$H(w_t, 7\pi/12) = 0 \tag{4.10}$$

$$H(w_t, 9\pi/12) = 0 \tag{4.11}$$

$$H(w_t, 11\pi/12) = 0 \tag{4.12}$$

Pour aplatir le spectre partout à partir de $w_v = \pi/3$, on a imposé les huit contraintes précédentes, ce qui donne seize équations.

$$H(0, \pi/12) = 6 \tag{4.13}$$

$$H^{|}(0, \pi/12) = 6 \tag{4.14}$$

$$H(0, \pi/6) = 4 \tag{4.15}$$

Ces trois dernières contraintes sont imposées pour avoir un gain parfait autour de l'origine.

Le choix des contraintes est très important. Afin d'éviter la dépendance et l'incohérence entre les équations, on commence par les contraintes prioritaires et on impose d'autre contraintes au système d'équations au fur et à mesure. En examinant le dessin 3D de la réponse en fréquence des coefficients obtenus, on a une idée de la direction à prendre et celle de la contrainte qui doit être imposée.

La forme matricielle de ce système d'équations est:

AX = B

$$\Rightarrow X = A^{-1}B \tag{4.16}$$

À l'aide de Matlab, on a pu trouver les inconnues, i.e les 21 coefficients du filtre. Ces coefficients sont quantifiés en fractions de 512 comme le montre le tableau 4.4 .

$\frac{1}{512}$

a1	a2	a3	a4	a5	a7	a8	a9	a10	a11	
464	342	192	68	4	30	60	64	42	4	
b0	b1	b2	b4	b5	b6	b7	b8	b10	b11	b12
146	124	68	-52	-74	-64	-36	-11	-4	-14	-10

Tableau 4.4 Les coefficients du filtre V-T à polyphase

En substituant les coefficients trouvés dans l'équation (4.2), on obtient la réponse fréquentielle H(w_t,w_v) en 3D de la figure 4.7. Il est bon de noter que la caractéristique «passe tout» est respectée, au moins pour les basses fréquences verticales.

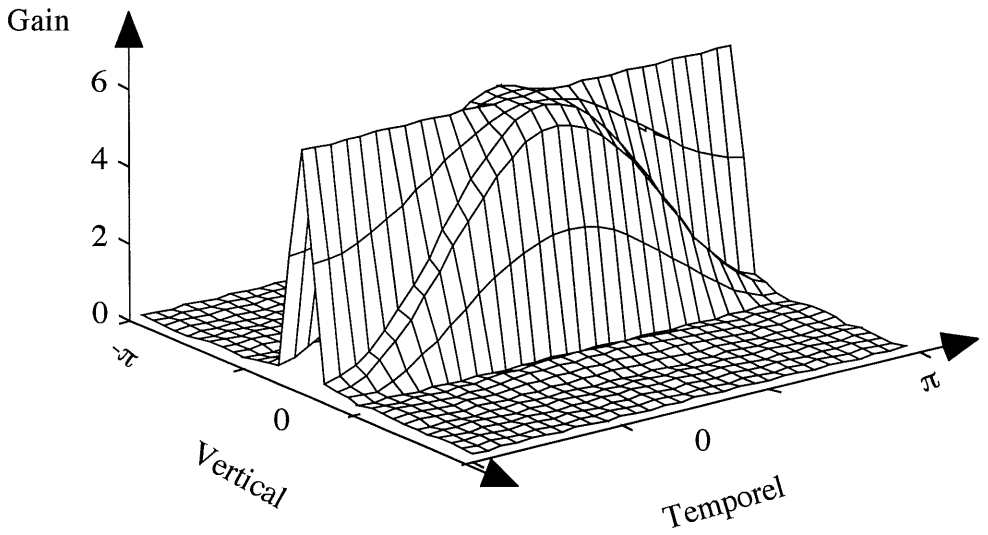


Figure 4.7 La présentation 3D du filtre V-T trouvé

Le contour et les atténuations du filtre V-T avec la fréquence verticale variant de -787,5/2 cph (cycle par hauteur) à 787,5/2 cph et avec la fréquence temporelle variant de -60/2 Hz à 60/2 Hz sont illustrés en figure 4.8.

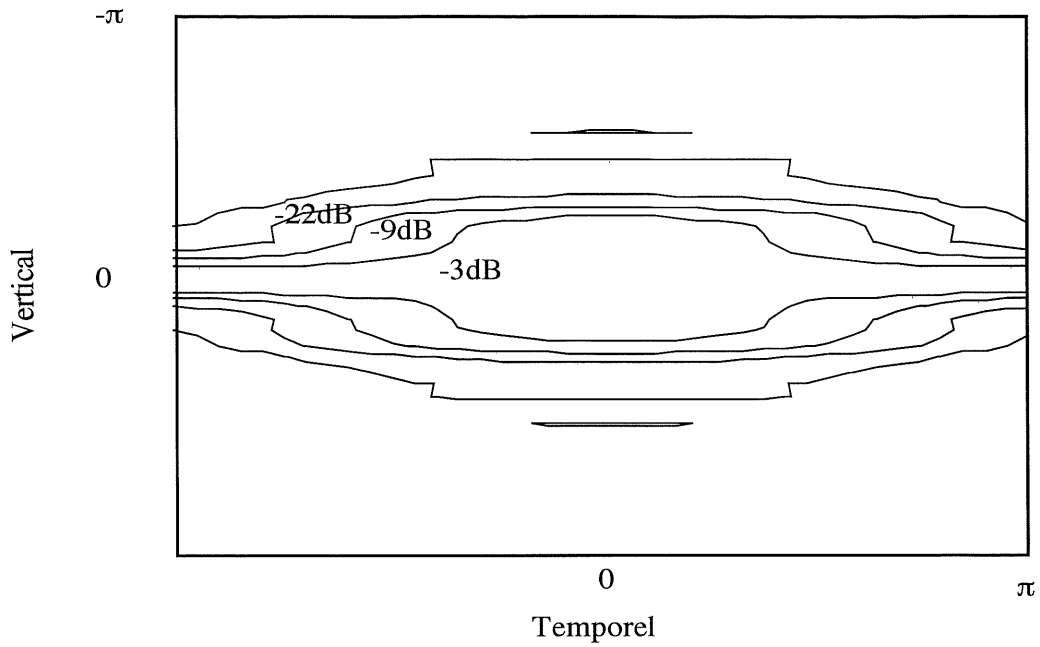


Figure 4.8 Contours isopotentiels du filtre V-T polyphase

4.3 Conclusion

De façon pratique, le problème majeur de conversion provient de l'entrelacement du signal; l'approche indirecte est conditionnée par la performance du doubleur de lignes.

Quant à la conversion directe, elle est basée sur un filtre V-T polyphase dont la phase centrale est la plus favorable. En réarrangeant le filtre polyphase, on peut voir que cette phase n'est pas autre chose que le filtre V-T du doubleur de lignes. Cela veut dire que si on n'arrive pas à obtenir une bonne image avec le filtre V-T du doubleur de lignes, on n'arrivera jamais à en obtenir une avec un filtre polyphase. La performance d'une telle conversion est toujours inférieure à celle d'un doubleur de lignes non linéaire et adaptatif en fonction de la direction du contour.

En terminant, on aurait dû concevoir un filtre polyphase adaptatif, mais une telle conception augmenterait énormement la complexité et risquerait d'introduire beaucoup d'erreurs. Il nous paraît fort avantageux d'utiliser la méthode du doubleur de lignes du chapitre 3 suivie par deux interpolations verticale et horizontale appropriées.

5

EXPÉRIMENTATIONS, RÉSULTATS ET COMPARAISONS

Nous avons utilisé le langage C^{++} en tant qu'outil de programmation, ce qui nous a permis de simuler notre système sur l'ordinateur. La simulation est faite sur des images critiques, statiques et dynamiques. Les résultats obtenus sont présentés dans ce chapitre.

Un algorithme a été développé pour simuler le traitement.

5.1 L'algorithme de simulation

a-Décomposition et dé-entrelacement de l'image originale

D'abord chaque image à traiter a été décomposée en trois composantes Y, U, V. Ensuite on a dé-entrelacé les champs pairs et les champs impairs. Enfin, on a suréchantillonné ces champs en ajoutant des lignes noires entre les lignes d'image.

b-Détection de contours

La détection de contour a été faite séparément pour chaque composante d'un champ. On a calculé la variation de niveau de gris pour chacune des cinq directions choisies puis on a fait une comparaison quantitative pour trouver celle de plus faible valeur. La direction donnant la plus faible valeur de variation est celle qui englobe le contour.

c-Fortification de la décision

Ici, on a cherché à consolider les vrais contours et à éliminer les faux contours. La technique de renforcement des décisions séquentielles a été appliquée selon la direction du contour. En d'autres termes, il faut avoir au moins deux points sur trois inclinés dans la même direction pour que la direction du point au milieu soit retenue. Sinon, on retient la direction verticale.

d-Interpolation selon le contour

À chaque point à interpoler, on a adapté le filtre V-T dans sa direction de contour. L'adaptation des filtres V-T de base selon les directions du contour a produit cinq filtres distincts. Chaque point a été interpolé avec son filtre approprié, la priorité étant accordée à la direction verticale.

e-Détection de mouvement et adaptation spatio-temporelle

La détection de mouvement est basée sur la différence des niveaux de gris de deux champs adjacents. Si cette différence est inférieure à un certain seuil M_1, on considère que la zone est localement immobile. Si cette différence est supérieure à un certain seuil M_2 on utilise l'interpolation selon le contour. Si cette différence se situe entre M_1 et M_2, on prend la contribution des deux interpolations selon une relation linéaire.

5.2 Résultats et comparaisons

À la fin du présent chapitre, nous allons présenter les images prises directement à l'écran d'une station Sun de l'Université Sherbrooke. Elles ont été simulées par notre algorithme et celui de Tsarcriq. Elles sont comparées aux images originales que l'on montre à titre de référence.

5.2.1 Cas des images dynamiques

Ici, nous traitons une séquence d'images «Flower» très critique qui est entièrement en mouvement. Ainsi, les figures citées ci-dessous et présentées à la fin du chapitre montrent les trois composantes Y, U, V des images traitées avec différents algorithmes:

- la première image entrelacée «Flower» (sans traitement) (Figure 5.1);
- la deuxième image entrelacée «Flower» (sans traitement) (Figure 5.2);
- l'image progressive «Flower» du champ pair traitée avec l'algorithme de Tsarcriq (Figure 5.3);
- l'image progressive «Flower» du champ impair traitée avec l'algorithme de Tsarcriq (Figure 5.4);

- l'image progressive «Flower» du champ pair traitée avec notre algorithme (Figure 5.5);
- l'image progressive «Flower» du champ impair traitée avec notre algorithme (Figure 5.6).

On peut facilement observer une nette amélioration du contour du tronc et de celui des branches d'arbre sur l'image simulée par notre algorithme par rapport aux autres images. De plus, on note moins de papillotement dans les fleurs.

5.2.2 Cas des images statiques

Dans ce cas, nous utilisons une image très critique «Poster» qui contient beaucoup de détails avec de très hautes fréquences verticales. Les trois composantes Y, U, V des images traitées avec différents algorithmes sont représentées sur les figures de la fin du chapitre:

- l'image entrelacée «Poster» (sans traitement) (Figure 5.7);
- l'image progressive «Poster» du champ pair traitée avec l'algorithme de Tsarcriq (Figure 5.8);
- l'image progressive «Poster» du champ impair traitée avec l'algorithme de Tsarcriq (Figure 5.9);
- l'image progressive «Poster» du champ pair traitée avec notre algorithme (Figure 5.10);
- l'image progressive «Poster» du champ impair traitée avec notre algorithme (Figure 5.11).

Comme il n'y a pas de mouvement dans le «Poster», l'image entrelacée ne présente aucun défaut. Pourtant, l'image issue de notre algorithme (même sans détecteur de mouvement) est quasiment identique à celle-ci.

5.2.3 Cas des images TVHD

La conversion aux formats TVHD s'est faite à partir du signal progressif de notre doubleur de lignes. Les figures suivantes (à la fin du chapitre) montrent quelques images TVHD obtenues:

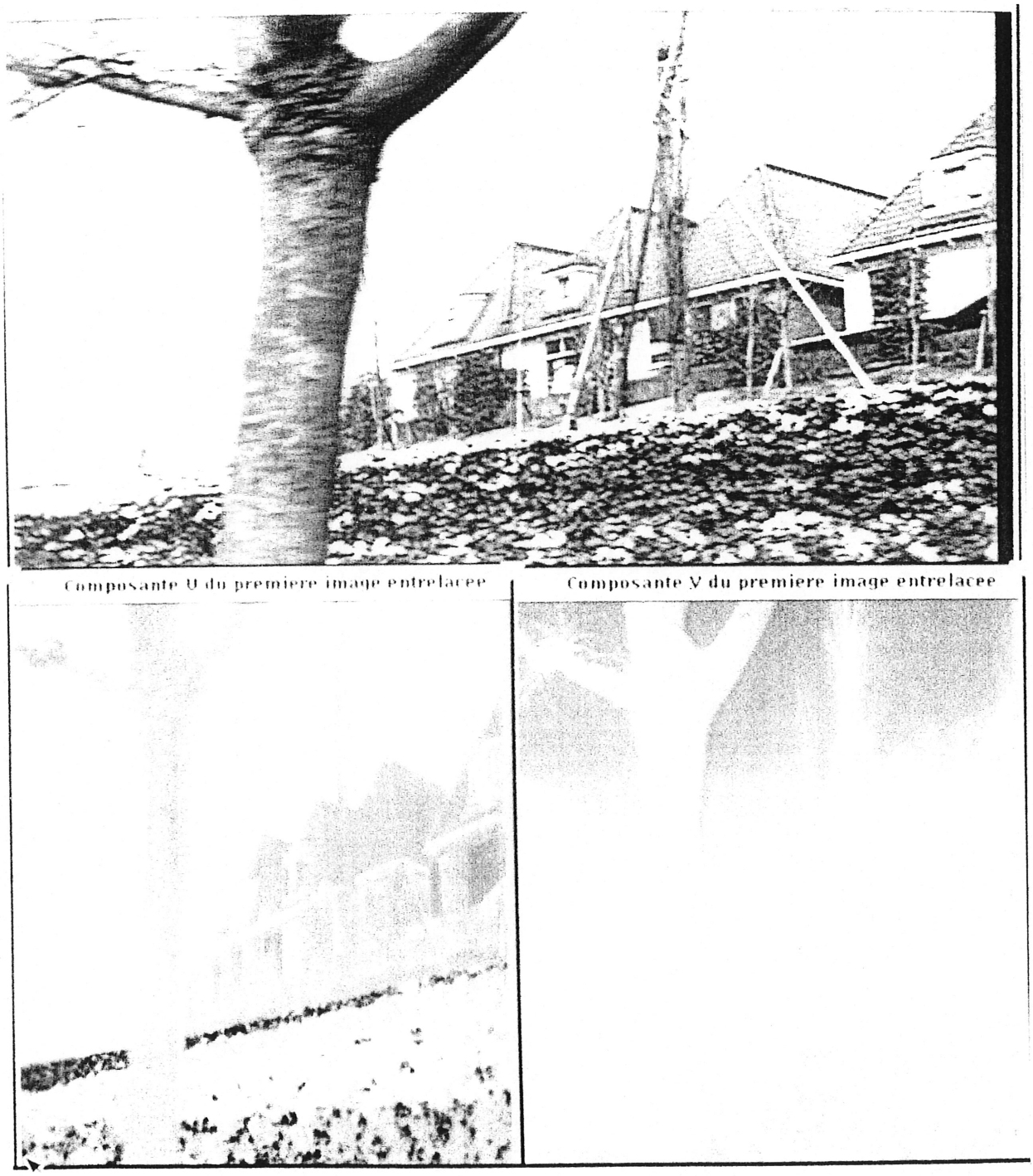
- format TVHD d'image dynamique à 1050 lignes (Figure 5.12);
- format TVHD d'image statique à 1050 lignes (Figure 5.13);
- format TVHD d'image dynamique à 787.5 lignes (Figure 5.14).

5.3 Conclusion

Compte tenu de la qualité de la prise des photos, les images montrées ne sont pas parfaitement claires en ce qui nous concerne. Cependant, elles ont pu démontrer la suffisance du doubleur de lignes proposé.

L'évaluation de la qualité perceptuelle d'images traitées a été faite par des experts de ce domaine. Des tests subjectifs semblent offrir le meilleur moyen pour évaluer la qualité des images dynamiques. Quant aux images statiques, on a préféré une comparaison quantitative. Le résultat est impeccable avec le détecteur de mouvement et très acceptable sans détecteur de mouvement.

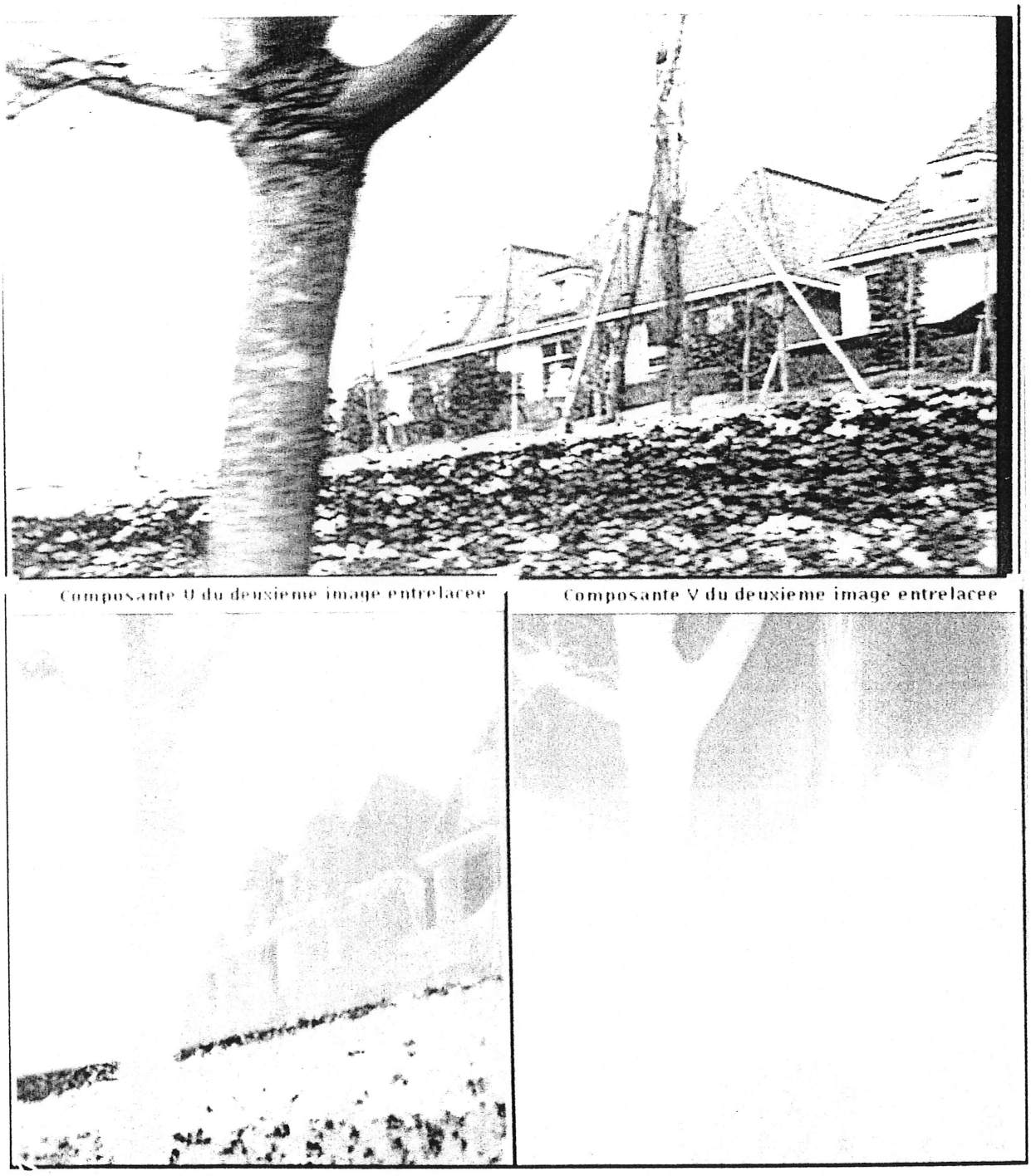
a



b c

Figure 5.1 La première image entrelacée «Flower» (sans traitement)
a) la composante Y b) la composante U c) la composante V

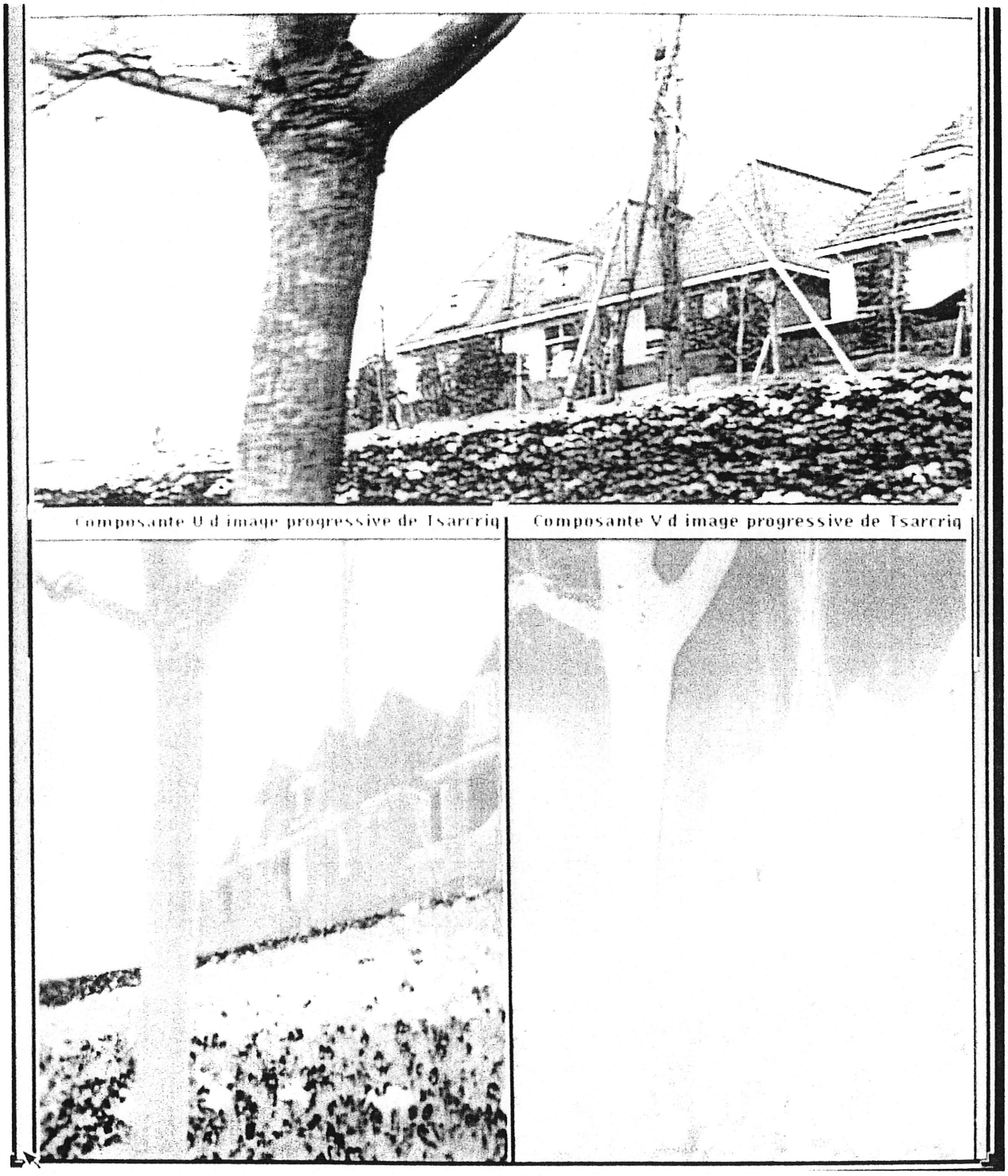
a



b c

Figure 5.2 La deuxième image entrelacée «Flower» (sans traitement)
a) la composante Y b) la composante U c) la composante V

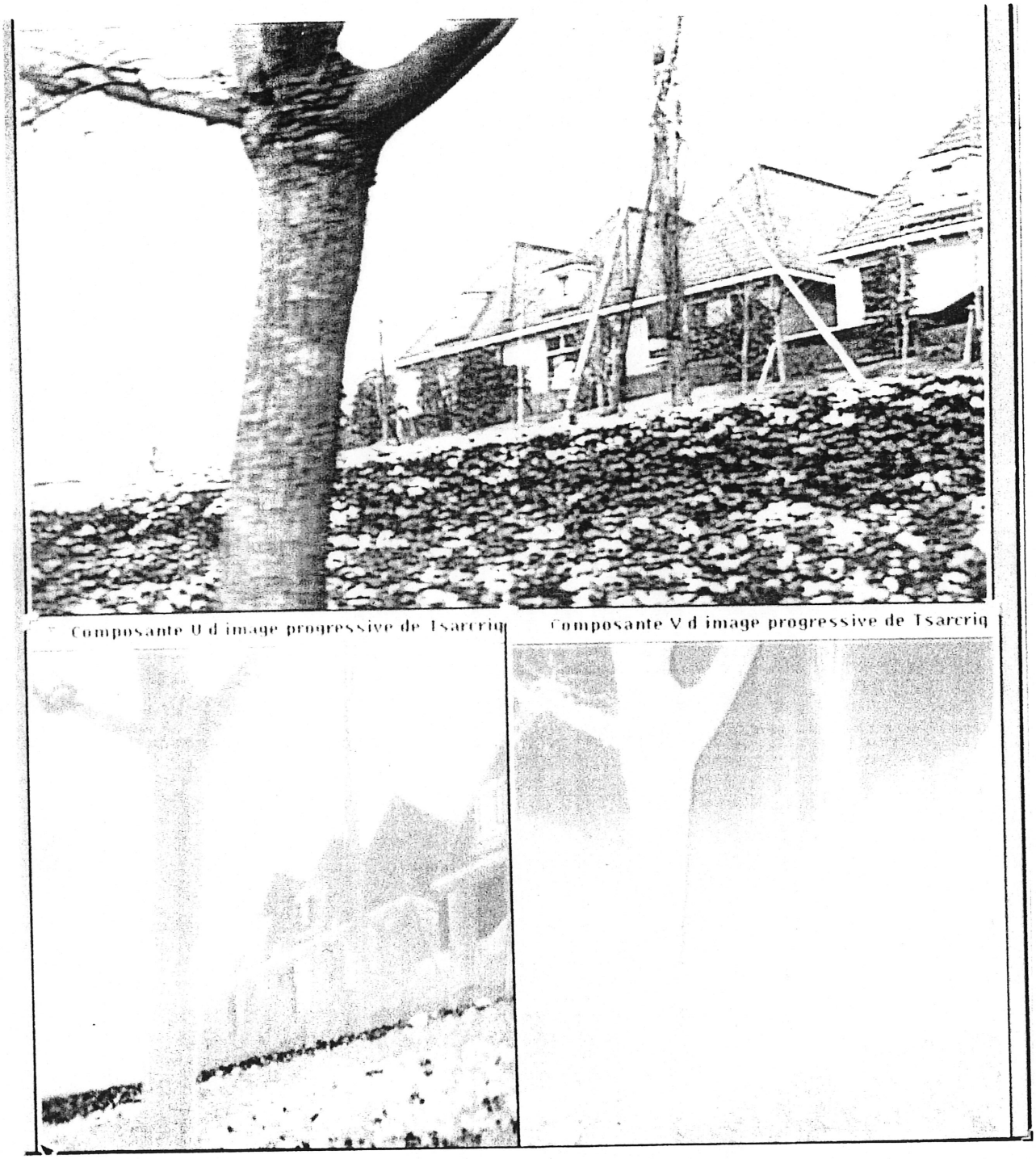
a



b c

Figure 5.3 L'image progressive «Flower» du champ pair traitée avec l'algorithme de Tsarcriq a) la composante Y b) la composante U c) la composante V

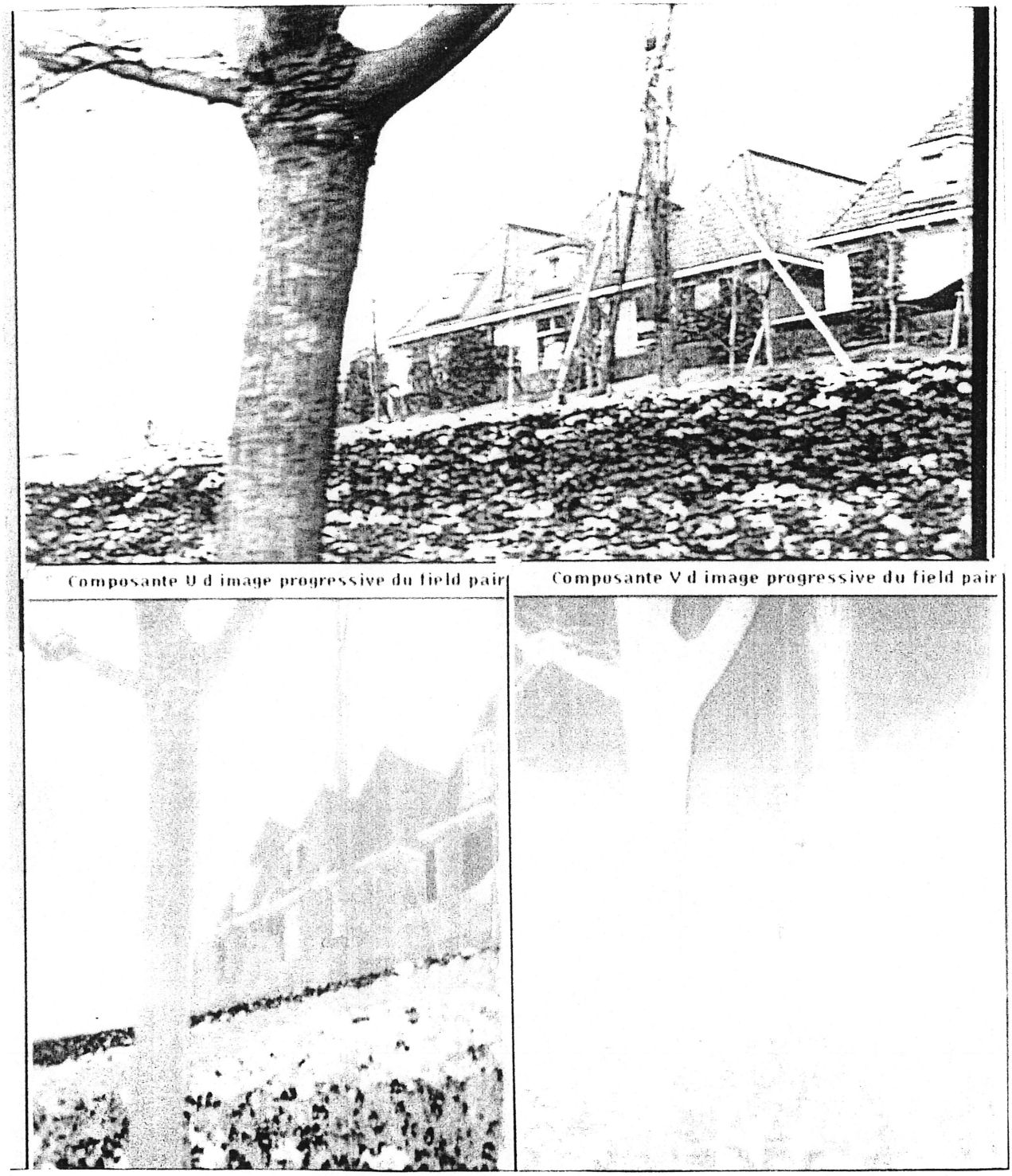
a



b c

Figure 5.4 L'image progressive «Flower» du champ impair traitée avec l'algorithme de Tsarcri a) la composante Y b) la composante U c) la composante V

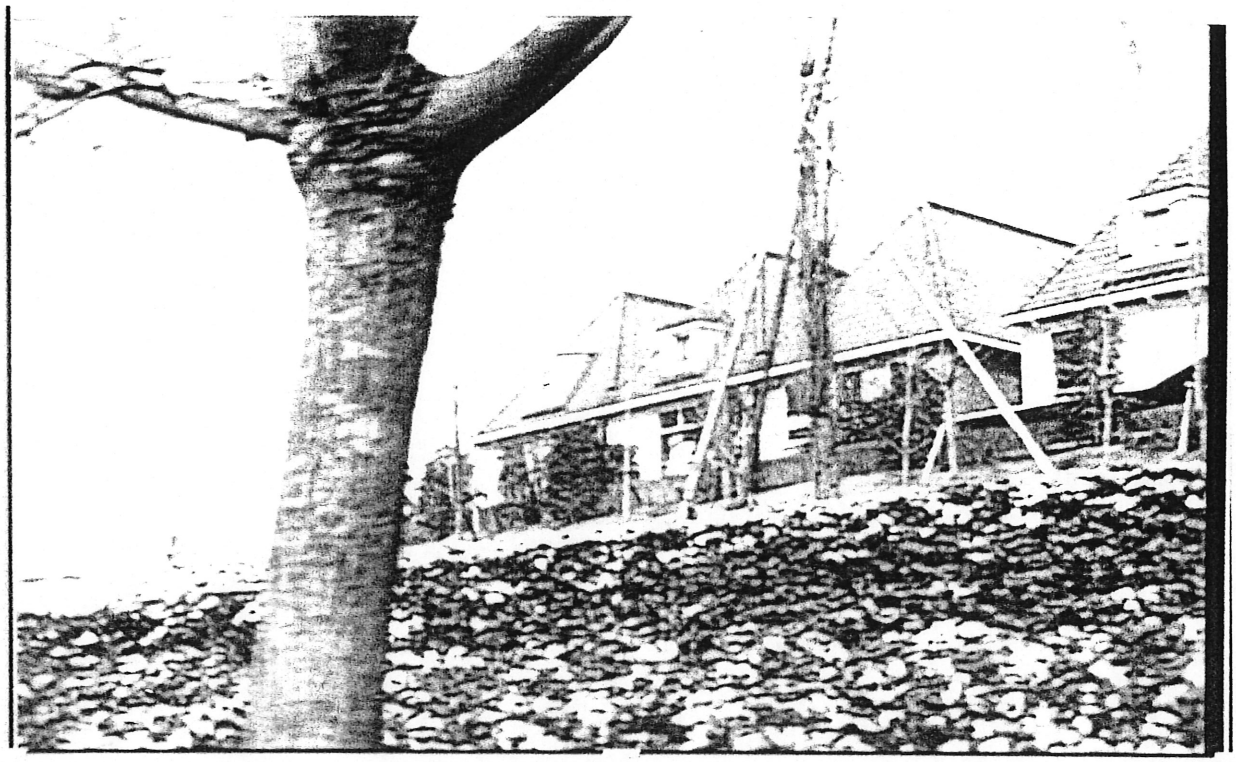
a

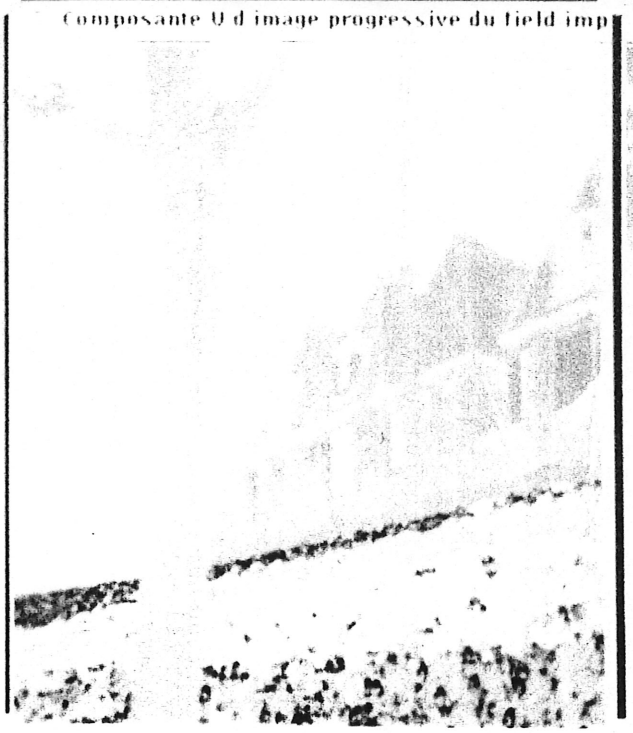


b c

Figure 5.5 L'image progressive «Flower» du champ pair traitée avec notre algorithme
a) la composante Y b) la composante U c) la composante V

a





b

c

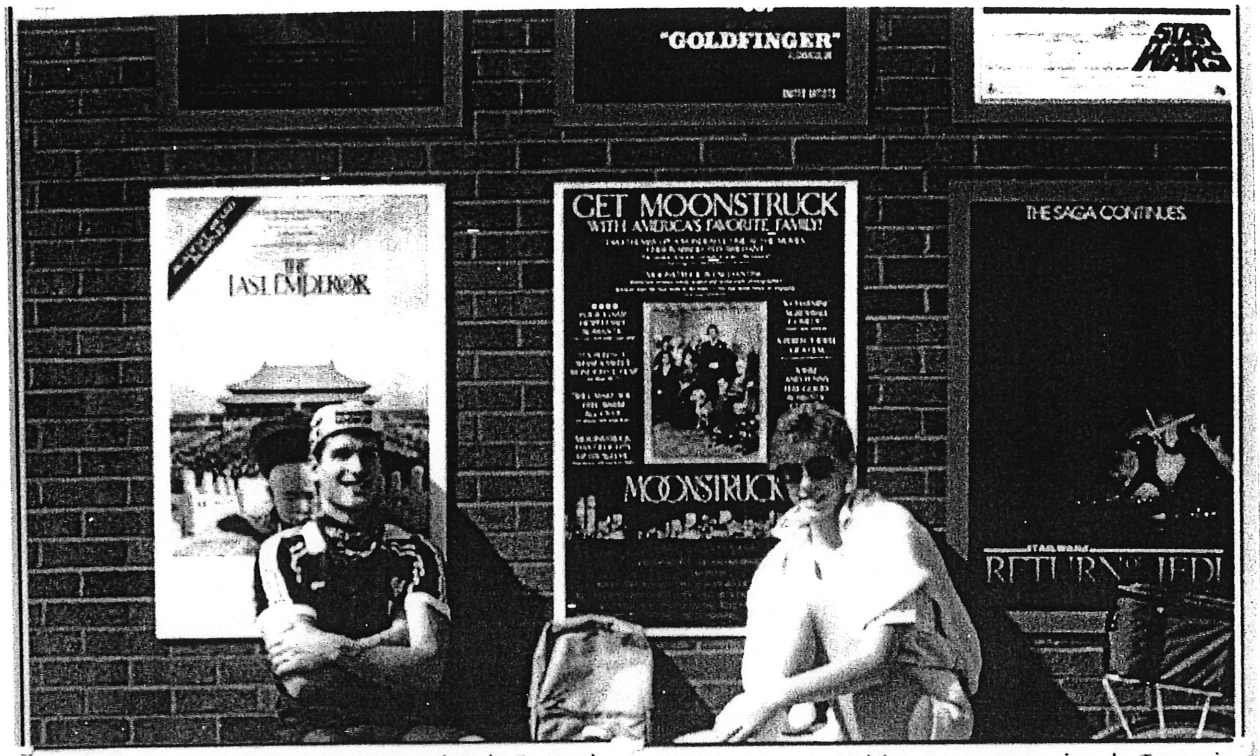
Figure 5.6 L'image progressive «Flower» du champ impair traitée avec notre algorithme
a) la composante Y b) la composante U c) la composante V

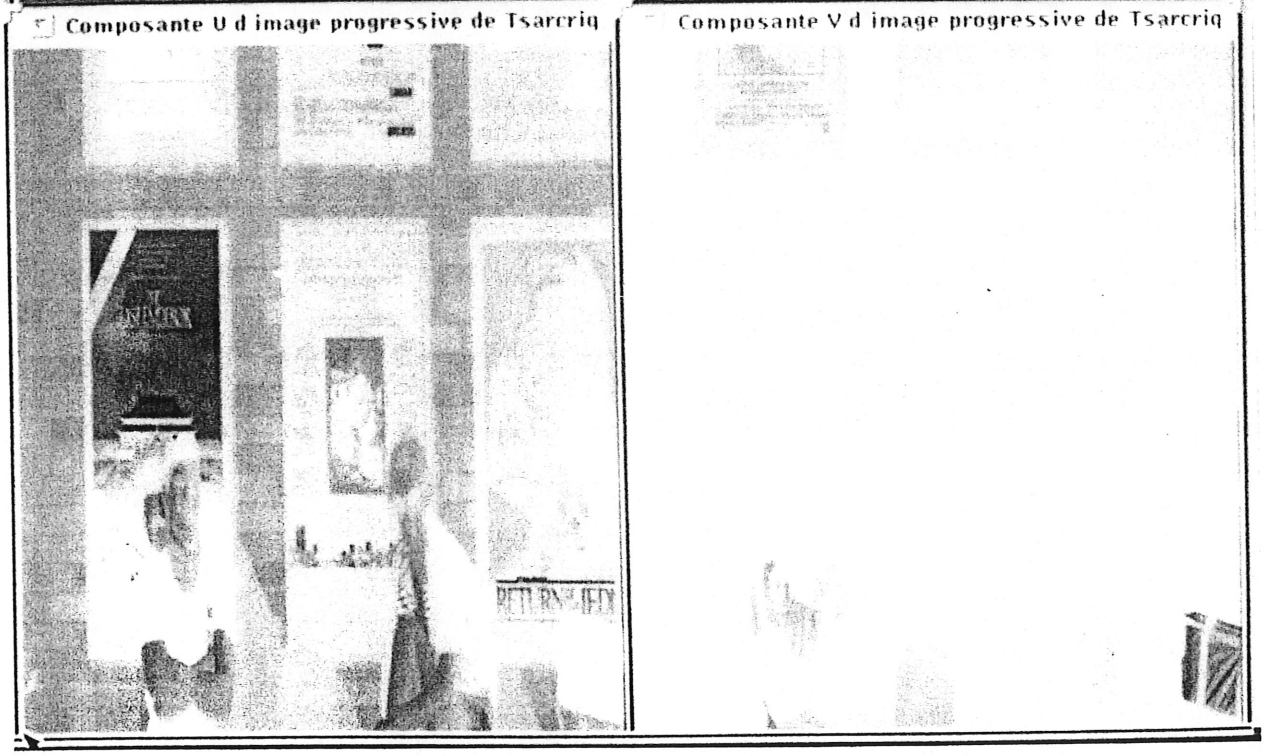
a

b c

Figure 5.7 L'image entrelacée «Poster» (sans traitement)
a) la composante Y b) la composante U c) la composante V

a

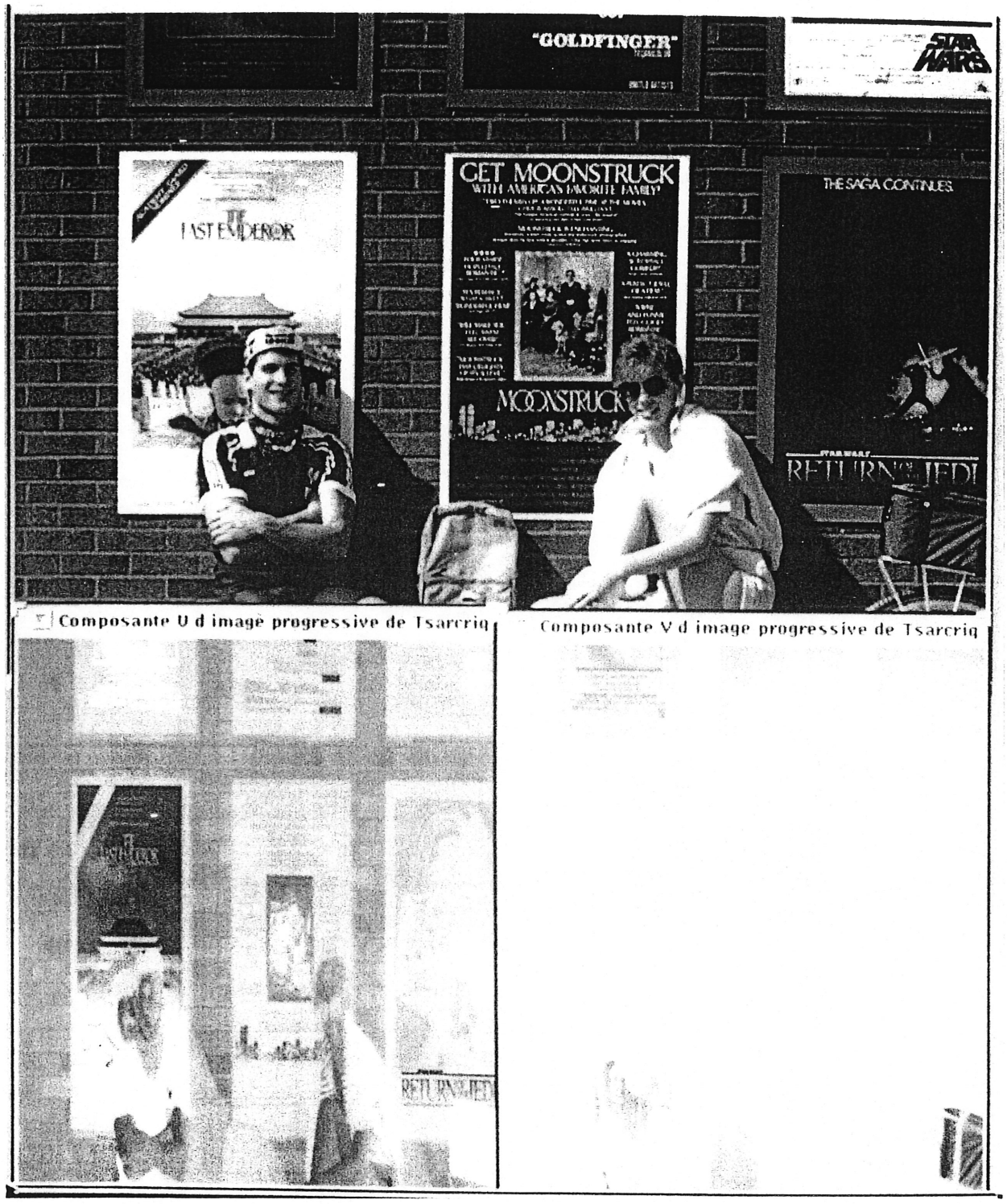




b c

Figure 5.8 L'image progressive «Poster» du champ pair traitée avec l'algorithme de Tsarcriq a) la composante Y b) la composante U c) la composante V

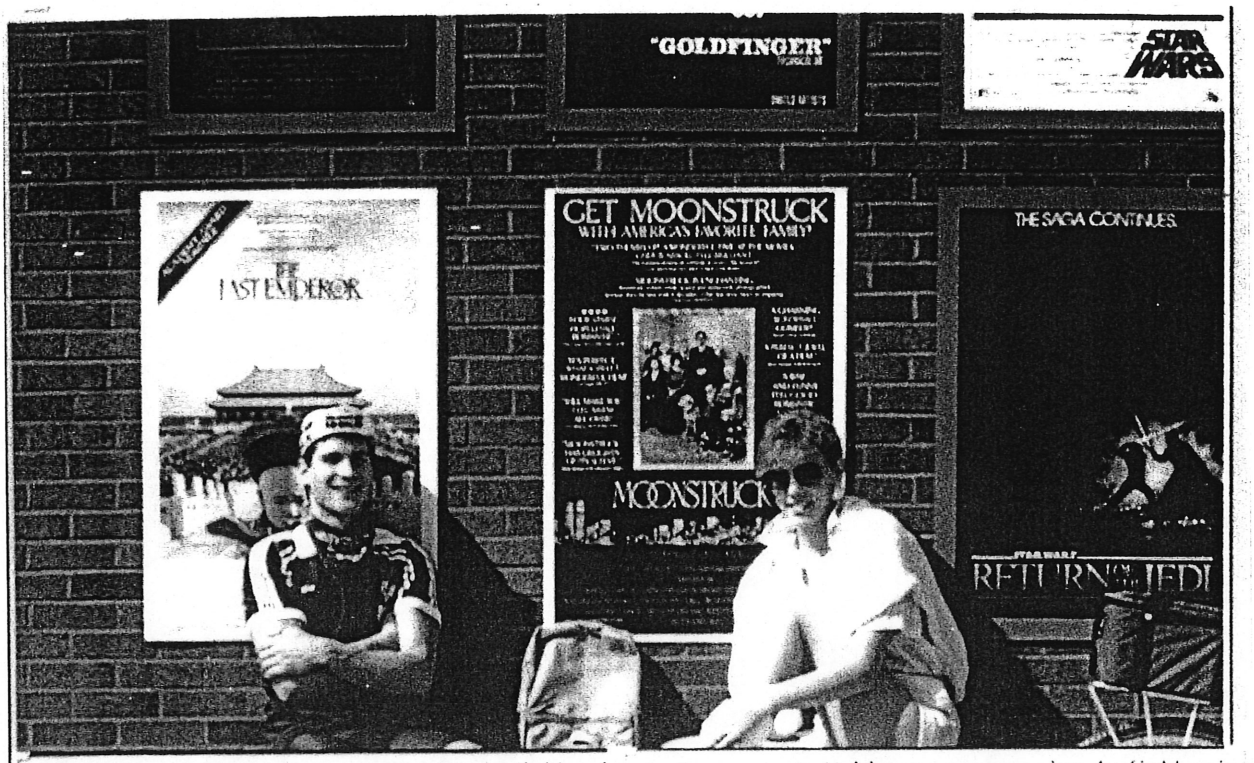
a

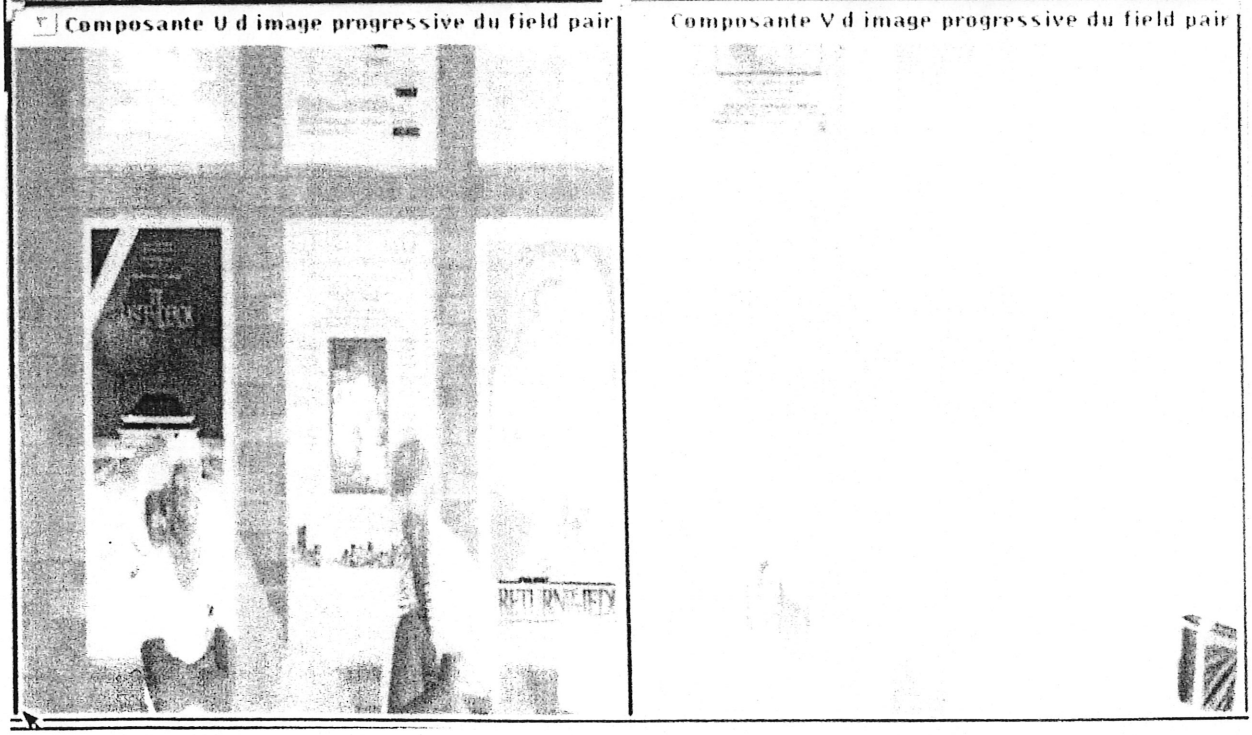


b c

Figure 5.9 L'image progressive «Poster» du champ impair traitée avec l'algorithme de Tsarcriq a) la composante Y b) la composante U c) la composante V

a

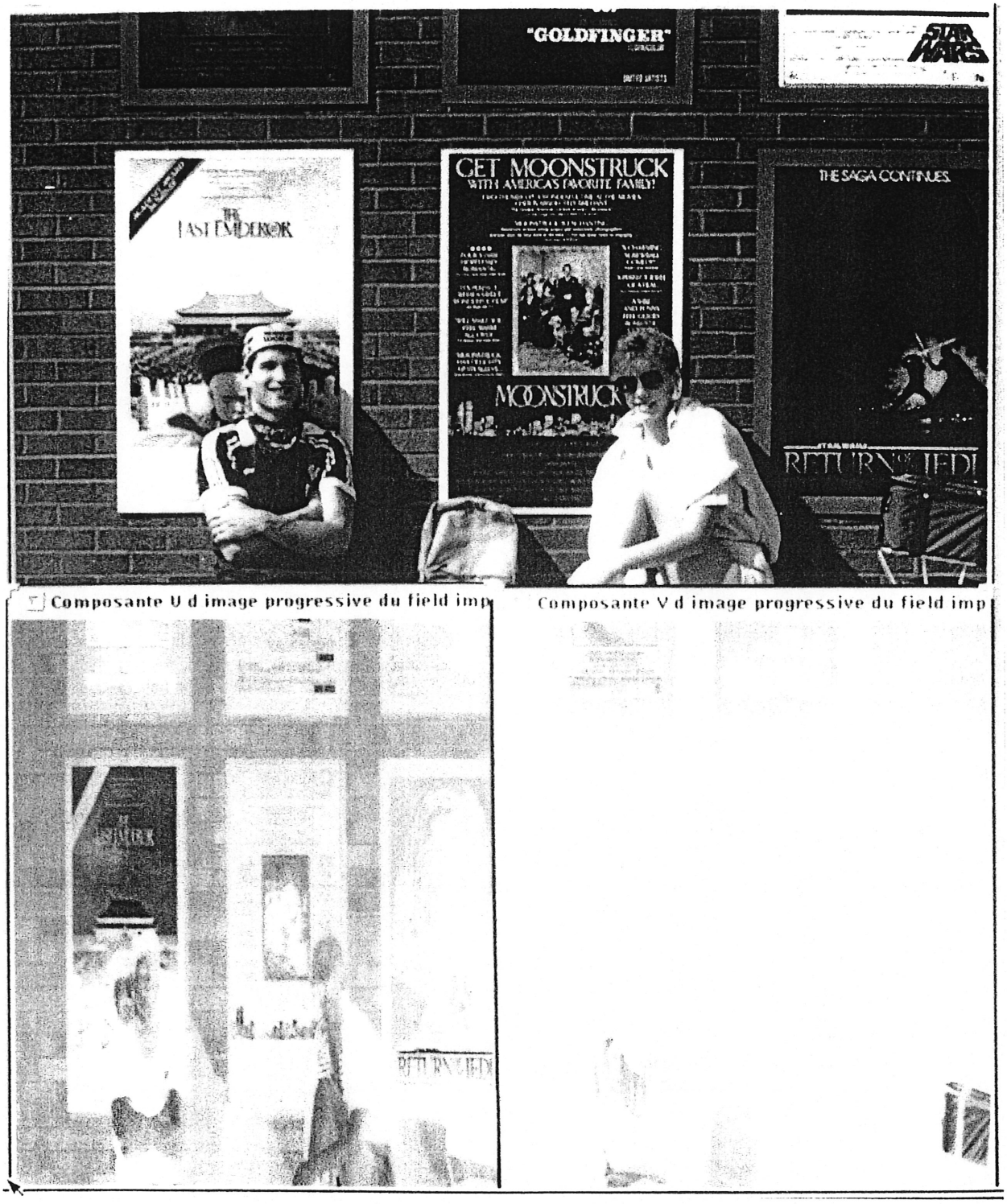




b c

Figure 5.10 L'image progressive «Poster» du champ pair traitée avec notre algorithme
a) la composante Y b) la composante U c) la composante V

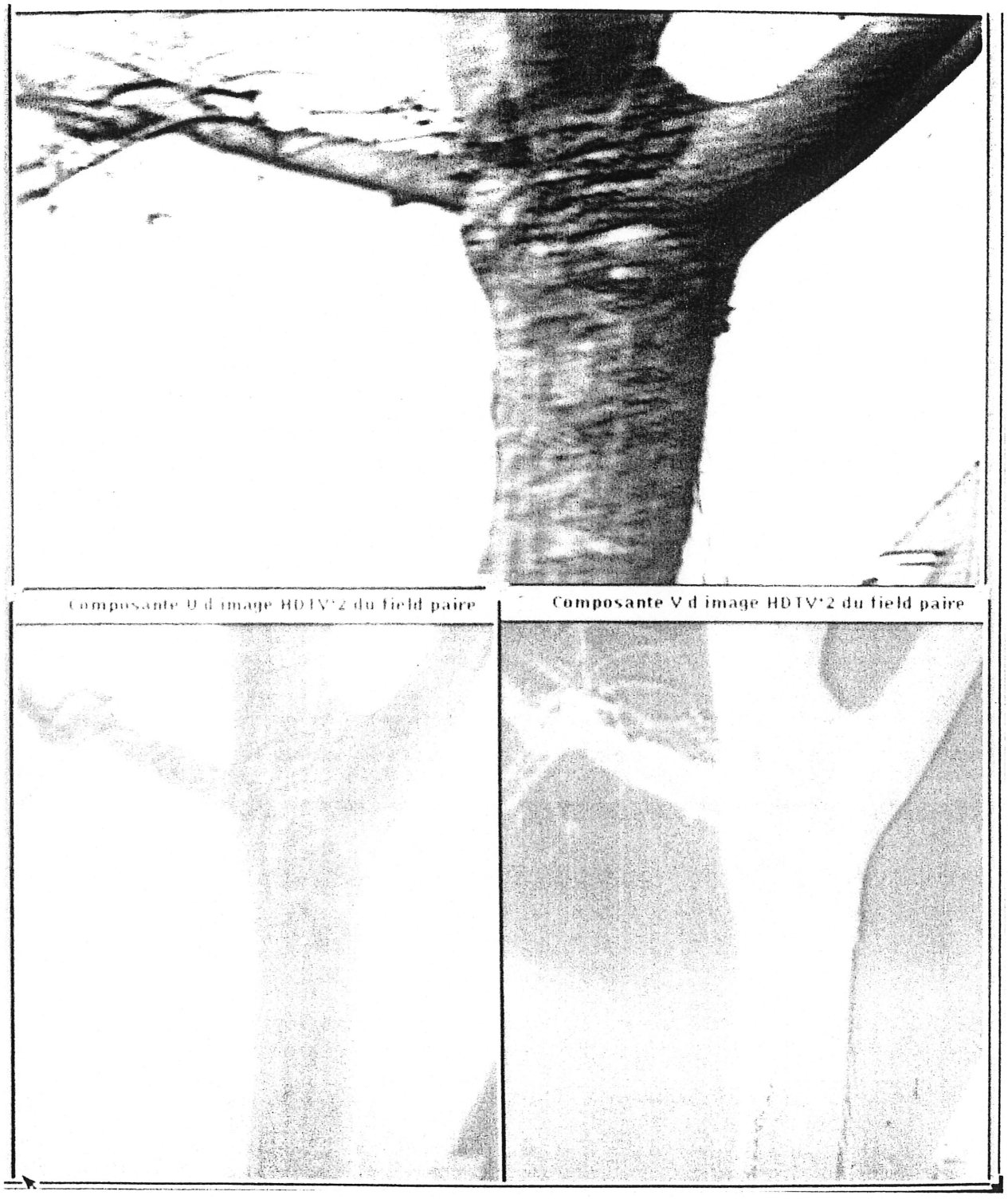
a



b c

Figure 5.11 L'image progressive «Poster» du champ impair traitée avec notre algorithme
a) la composante Y b) la composante U c) la composante V

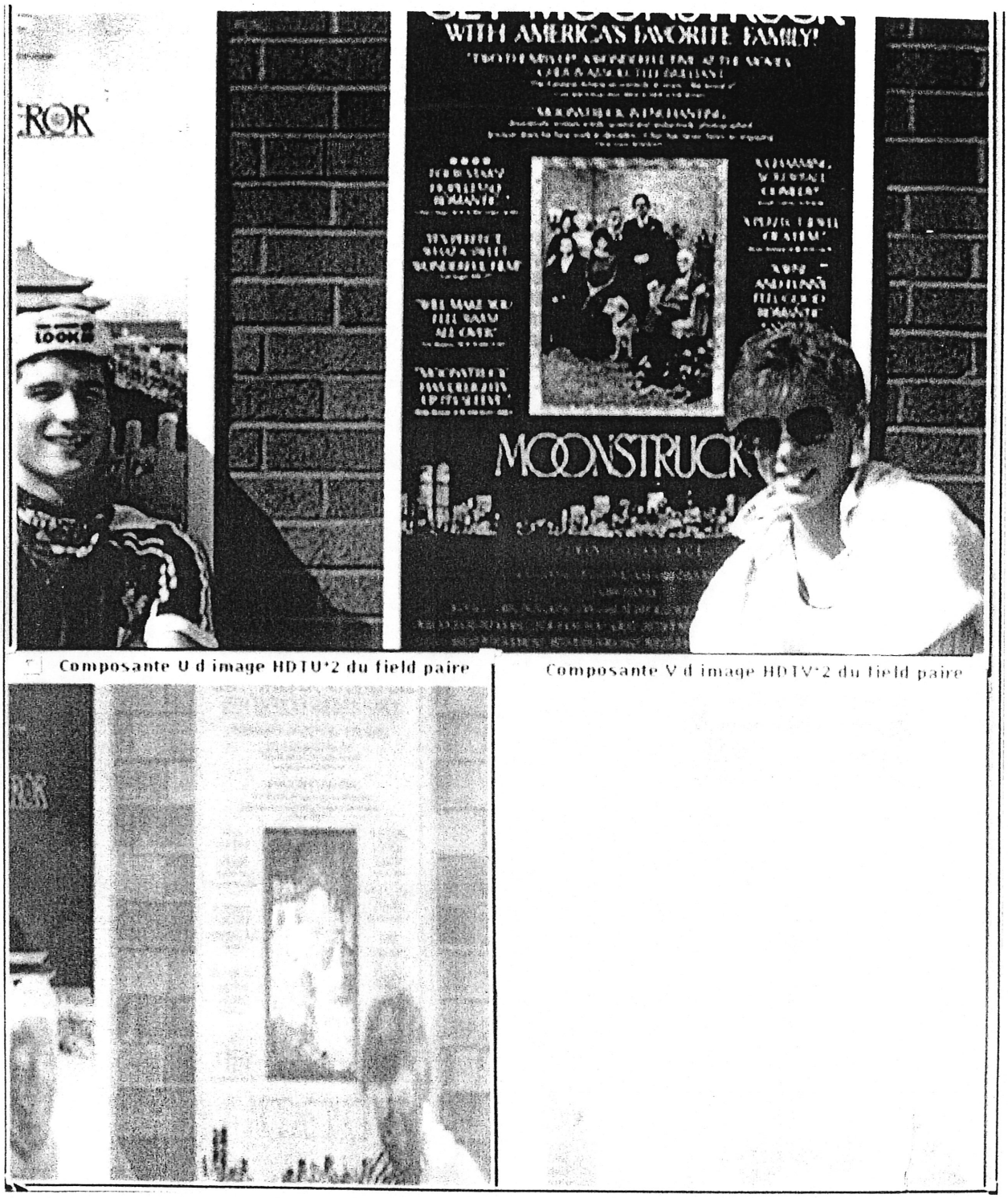
a



b c

Figure 5.12 Format TVHD d'image dynamique «Flower» à 1050 lignes
a) la composante Y b) la composante U c) la composante V

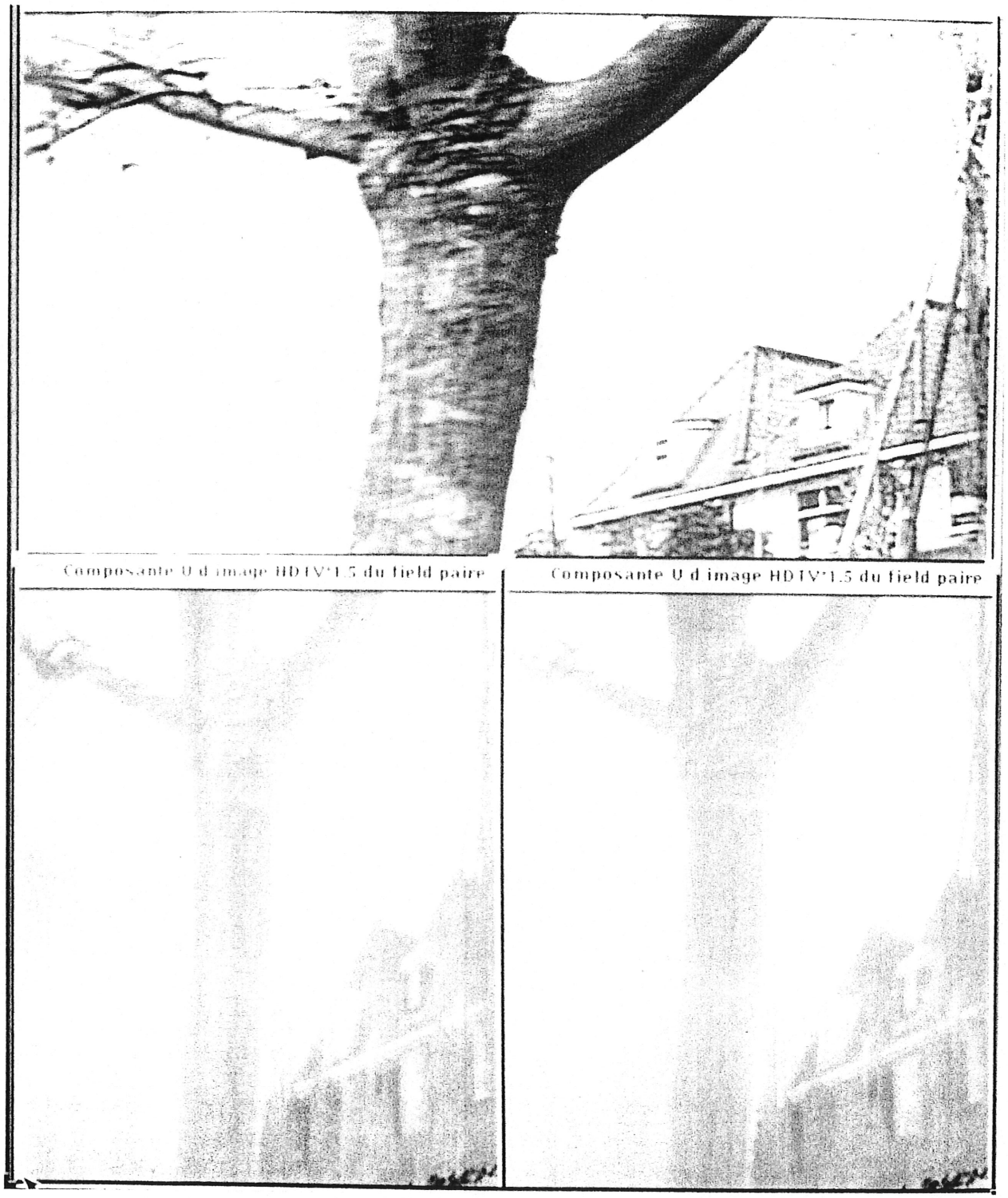
a



b c

Figure 5.13 Format TVHD d'image statique «Poster» à 1050 lignes
a) la composante Y b) la composante U c) la composante V

a



b c

Figure 5.14 Format TVHD d'image dynamique «Flower» à 787,5 lignes
a) la composante Y b) la composante U c) la composante V

CONCLUSION

Ce mémoire a couvert plusieurs approches de conversion du signal entrelacé en signal progressif. L'étude des techniques de conversion existantes a permis de proposer un nouveau système efficace, simple, modulaire et réalisable.

Le système proposé se distingue par l'utilisation de techniques des moyennes et de renforcement des décisions séquentielles dans la détection de contour. De plus, l'adaptation des filtres spatio-temporels dans la direction du contour réduit l'effet d'escalier dans les contours inclinés. Par conséquent, l'image résultante du système proposé est beaucoup plus nette que l'originale et même que celles produites par les autre autres approches considérées. Elle offre une très bonne résolution verticale avec un défaut résiduel minime.

Il faut souligner que l'algorithme developpé a été testé et les images produites ont été visualisées en temps réel au laboratoire CRC à Ottawa. Sa performance et sa simplicité y ont été reconnues et appréciés par des scientifiques en traitement d'images du CRC. Ajoutons que sa réalisation est possible avec des circuits intégrés à très grande échelle.

De plus, une version améliorée de cet algorithme est exploitée commercialement par la compagnie Miranda Technologies inc., sous le nom de produit Quartz. Sa version logicielle pour une conversion de vidéo en film est utilisée présentement dans les laboratoires de Buzz Vidéo, Softimage, etc.

En conclusion, la présente étude fut très enrichissante pour son auteur qui souhaite surtout qu'elle saura contribuer d'une façon ou d'une autre à faire avancer la recherche dans ce domaine.

ANNEXE A

A-1 PROGRAMME POUR CALCULER LES COEFFICIENTS ET DESSINER LE SPECTRE 3D DU FILTRE V-T POLYPHASE

```
clear
clc
clg
w=0;
p=pi;
w2=2*w;w3=3*w;w4=4*w;w5=5*w;w6=6*w;w7=7*w;w8=8*w;w9=9*w;w10=10*w;w11=11
*w;
w12=12*w;w13=13*w;w14=14*w;
A1=[cos(w);cos(w2);cos(w3);cos(w4);cos(w5);cos(w7);cos(w8);cos(w9);cos(w10);
cos(w11)]';
B1=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C1=[A1 B1];
A2=[0;0;0;0;0;0;0;0;0;0;0;]';
B2=[1;2;2;2;2;2;2;2;2;2;2;2;2;2]';
C2=[A2 B2];
w=p/3;
A3=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B3=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C3=[A3 B3];
A4=[0;0;0;0;0;0;0;0;0;0;0;]';
B4=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C4=[A4 B4];
w=p/2;
A5=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B5=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C5=[A5 B5];
A6=[0;0;0;0;0;0;0;0;0;0;0;]';
B6=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C6=[A6 B6];
w=5*p/6;
A7=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B7=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C7=[A7 B7];
A8=[0;0;0;0;0;0;0;0;0;0;0;]';
B8=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C8=[A8 B8];
w=p;
A9=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B9=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C9=[A9 B9];
```

```
A10=[0;0;0;0;0;0;0;0;0;0;0;]';
B10=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C10=[A10 B10];
w=p/12;
A11=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B11=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C11=[A11 B11];
A12=[sin(w);2*sin(w*2);3*sin(w*3);4*sin(w*4);5*sin(w*5);7*sin(w*7);8*sin(w*8);
9*sin(w*9);10*sin(w*10);11*sin(w*11)]';
B12=2*[0;sin(w);2*sin(w*2);4*sin(w*4);5*sin(w*5);6*sin(w*6);7*sin(w*7);
8*sin(w*8);10*sin(w*10);11*sin(w*11);12*sin(w*12);13*sin(w*13);14*sin(w*14)]';
C12=[A12 B12];

w=5*p/12;
A13=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B13=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C13=[A13 B13];
A14=[0;0;0;0;0;0;0;0;0;0;0;]';
B14=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C14=[A14 B14];
w=7*p/12;
A15=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B15=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C15=[A15 B15];
A16=[0;0;0;0;0;0;0;0;0;0;0;]';
B16=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C16=[A16 B16];
w=9*p/12;
A17=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B17=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C17=[A17 B17];
A18=[0;0;0;0;0;0;0;0;0;0;0;]';
B18=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C18=[A18 B18];
w=11*p/12;
A19=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
cos(w*10);cos(w*11)]';
B19=[0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;0 ;]';
C19=[A19 B19];
A20=[0;0;0;0;0;0;0;0;0;0;0;]';
B20=[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C20=[A20 B20];
w=p/6;t=0;
A21=[cos(w);cos(w*2);cos(w*3);cos(w*4);cos(w*5);cos(w*7);cos(w*8);cos(w*9);
```

```
cos(w*10);cos(w*11)]';
B21=cos(t)*[1 ;2*cos(w);2*cos(w*2);2*cos(w*4);2*cos(w*5);2*cos(w*6);2*cos(w*7);
2*cos(w*8);2*cos(w*10);2*cos(w*11);2*cos(w*12);2*cos(w*13);2*cos(w*14)]';
C21=[A21 B21];

D=[C1;C3;C5;C7;C9;C11;C13;C15;C17;C19;C2;C4;C6;C8;C10;C12;C14;C16;C18;C20;C21];
E=D(1:21,1:21);
e=det(E);
F=inv(E);
G=2*[640;-128;-128;-128;-128;640;-128;-128;-128;-128;0;0;0;0;0;0;0;0;0;0;636];
G1=G(1:21);
H=F*G1
c1=H(1,1);c2=H(2,1);c3=H(3,1);c4=H(4,1);c5=H(5,1);c6=H(6,1);c7=H(7,1);c8=H(8,1);c9=H(9
,1);c10=H(10,1);c11=H(11,1);c12=H(12,1);c13=H(13,1);c14=H(14,1);
c15=H(15,1);c16=H(16,1);c17=H(17,1);c18=H(18);c19=H(19);c20=H(20);c21=H(21);

H1=[ c21   0    c21
   c20   c10   c20
   c19   c9    c19
   0     c8    0
   c18   c7    c18
   c17   c6    c17
   c16   0     c16
   c15   c5    c15
   c14   c4    c14
   0     c3    0
   c13   c2    c13
   c12   c1    c12
   c11   512   c11
   c12   c1    c12
   c13   c2    c13
   0     c3    0
   c14   c4    c14
   c15   c5    c15
   c16   0     c16
   c17   c6    c17
   c18   c7    c18
   0     c8    0
   c19   c9    c19
   c20   c10   c20
   c21   0     c21];

H2=H1(1:25,1:2);

W=zeros(40,40);
W(1:25,1:3)=(1/512)*H1;
S=fft2(W);
S=fftshift(abs(S));
%S=20*log10(S/6);
V=[-3,-9,-18,-36]
contour(S,V)
mesh(S)
```

A-2 PROGRAMME POUR CALCULER LES COEFFICENTS DES FILTRES HORIZONTAL ET VERTICAL

```
%filtre de la conversion horizontale ou verticale
clc
clg
clear
%donnee

Ap=0.05;Aa=40;U=8;D=3;
fbande=6.75e6;
fp=5.50e6;
fa=8.00e6;
fs=2*U*6.75e6;
Ts=1/fs;
%debut
∂1=(10^(.05*Ap)-1)/(10^(.05*Ap)+1);
∂2=10^(-.05*Aa);
∂=min(∂1,∂2);
A=-20*log10(∂);
if(A<21) alpha=0;end
if(A>=21 & A<=50) alpha=.58417*(A-21)^.4+.07886*(A-21);end
if(A>50) alpha=.1102*(A-8.7);end
if(A<=21) D=.9222;end
if(A>21) D=(A-7.95)/14.36;end
Bt=2*pi*abs(fa-fp);
Ωc=1/2*2*pi*(fa+fp);
wc=Ωc*Ts;
wc/pi
pause
m=(2*pi/Ts)*(D/Bt);
M=ceil(m);
if(rem(M,2)==0) M=M+1;end
M
i=0;
%fenetre de kaiser
for n=-(M-1)/2:(M-1)/2
    i=i+1;
   beta=alpha*sqrt(1-((2*n)/(M-1))^2);
   Ia=0;
   Ib=0;
   for k=0:8
     Ia=Ia+(1/fact(k)*(alpha/2)^k)^2;
     Ib=Ib+(1/fact(k)*(beta/2)^k)^2;
   end
 WK(i)=Ib/Ia;
end
% Calculer les resultats de la réponse idéale hD(n)
hD=[];
i=0;
for n=-(M-1)/2:(M-1)/2
   i=i+1;
   if  n==0;
```

```
    hD(i)=wc/pi;
  else
    hD(i)=sin(n*wc)/(n*pi);
  end
end
h=hD.*WK;
save coef_hor
[H,w]=freqz(h,1,512);
plot(0:fs/2/512:fs/2-fs/2/512,20*log10(abs(H)))
```

Les coefficients du filtre horizontale

$\frac{1}{128}$

a0	a1	a2	a3	a4	a5	a6	a7	a8	a9	a10	a11	a12	a13	a14
128	125	115	100	81	60	38	17	0	-13	-22	-26	-25	-21	-15
a15	a16	a17	a18	a19	a20	a21	a22	a23	a24	a25	a26	a27	a28	a29
-7	0	6	11	13	13	11	8	4	0	-4	-6	-8	-8	-7
a30	a31	a32	a33	a34	a35	a36	a37	a38	a39	a40	a41	a42	a43	a44
-5	-2	0	2	4	5	5	4	3	1	0	-1	-2	-3	-3
a45	a46	a47	a48	a49	a50	a51	a52	a53	a54	a55	a56	a57	a58	a59
-2	-1	-1	0	1	1	1	1	1	0	0	-1	-1	0	0

ANNEXE B

PROGRAMME DE SIMULATION

```
#include <std.h>
#include <stdio.h>
#include <iostream.h>
#include <math.h>
#include <Ops/Ops.h>

#define paire 0
#define impaire 1
#define progr 1
#define facteur 2
#define nbfld 3


void showUse();
void doubler(char *fin, char *fout);
image *interpoler(const int a, const int b, const int c, char *fnm, char *t, int ind);
header *createheader(char *nm, definedType t, int seqlen, int intlc, int r, int c);
image *upsampling(int factor, int noframe, char *seqname, char *comp, int imp);
pixel getpixel(image &im, int x, int y);
void showpixel(const image *im, char *msg, int top, int left, int bottom, int right);
image *filtreVT(const image *past, const image *pres, const image *futur, int imp);
float SNR(const image *original, const image *im);
float abs(float Moy);

main(int argc, char **argv)
{

      char fentree[10], fsortie[10];

      if ((argc<3) || (argc>3)) showUse();

      strcpy(fentree, argv[1]);
      strcpy(fsortie, argv[2]);

      printf("\nfentree = %s\n", fentree);
      printf("fsortie = %s\n\n", fsortie);

      doubler(fentree, fsortie);

}


void showUse()
{
      printf("\n\n    Usage : doubler fichier1 fichier2 \n");
      printf("\n      fichier1 = fichier contenant image original.");
      printf("\n      fichier2 = fichier contenant image interpole.\n");
      printf("\n\n\n");
```

```
        exit(1);
}

image *interpoler(const int a, const int b, const int c, char *fnm, char *t, int ind)
{
        printf("interpoler composante %s\n", t);
        image *fld1 = upsampling(facteur, a, fnm, t, ind^impaire);
        image *fld2 = upsampling(facteur, b, fnm, t, ind^paire);
        image *fld3 = upsampling(facteur, c, fnm, t, ind^impaire);
        image *ret = filtreVT(fld1, fld2, fld3, ind^impaire);


        delete fld1;
        delete fld2;
        delete fld3;
        return ret;
}


void doubler(char *fin, char *fout)
{
        header *h;

        //calcul le nombre de sequence
        header tmph = header(fin);
        int nbseq;
        if (tmph.length==1)
                nbseq = 1;      //sans mouvement (une seule sequence)
        else
                nbseq = (tmph.length)*2-nbfld+1;   //avec mouvement
        printf ("nbseq = %d\n", nbseq);


        //filtrer
        image *resultat;
        int past=1, pres=1, fut=2, indice=0;
        if (nbseq==1) {
                if (strcmp(tmph.ftype,"Y")==0) {
                        resultat = interpoler(past, pres, past, fin, "Y", indice);
                        h = createheader(fout, Y, nbseq, progr,
                                        resultat->rows, resultat->cols);
                        h->write(fout);
                        resultat->append(fout, 1);

                        delete h;
                        delete resultat;
                }


                if (strcmp(tmph.ftype,"YUV")==0) {
                        resultat = interpoler(past, pres, past, fin, "Y", indice);
                        h = createheader(fout, YUV, nbseq, progr,
                                        resultat->rows, resultat->cols);
                        h->write(fout);
```

```
            resultat->append(fout, 1);

            delete h;
            delete resultat;


            resultat = interpoler(past, pres, past, fin, "U", indice);
            resultat->append(fout, 1);
            delete resultat;


            resultat = interpoler(past, pres, past, fin, "V", indice);
            resultat->append(fout, 1);
            delete resultat;

        }

}
else {
        for (int seq=1; seq<=nbseq; seq++) {
            printf("\nseq = %d   ", seq);
            printf("past = %d   ", past);
            printf("pres = %d   ", pres);
            printf("fut = %d   \n", fut);

            if (strcmp(tmph.ftype,"Y")==0) {
                resultat = interpoler(past, pres, fut, fin, "Y", indice);
                if (seq==1) {
                    h = createheader(fout, Y, nbseq,
                     progr, resultat->rows, resultat->cols);
                    h->write(fout);
                    delete h;
                }

                resultat->append(fout, seq);
                delete resultat;
            }



            if (strcmp(tmph.ftype,"YUV")==0) {
                resultat = interpoler(past, pres, fut, fin, "Y", indice);
                if (seq==1) {
                    h = createheader(fout, YUV, nbseq,
                          progr, resultat->rows, resultat->cols);
                    h->write(fout);
                    delete h;
                }
                resultat->append(fout, seq);
                delete resultat;


                resultat = interpoler(past, pres, fut, fin, "U", indice);
                resultat->append(fout, seq);
```

```
                    delete resultat;


                    resultat = interpoler(past, pres, fut, fin, "V", indice);
                    resultat->append(fout, seq);
                    delete resultat;

                }

                //recalculer past, pres, fut
                past = pres;
                pres = fut;
                fut = (past==pres) ? fut+1 : pres;
                indice = indice^1;

            } //end for
        } // end else
}

image *upsampling(int factor, int noframe, char *seqname, char *comp, int imp)
{
        int x, y, z=0;
        image *fld;
        image *im = new image(noframe, seqname, comp);
        image *ret = new image(im->rows*factor/2, im->cols);

        if (imp) {
                fld = im->deinterlace(0);
                for (y=0; y<ret->rows; y++) {
                        if ((y-factor/2)%factor==0) {
                                for (x=0; x<ret->cols; x++)
                                        ret->bitmap[y][x] = fld->bitmap[z][x];
                                z += 1;
                        }
                        else
                                for (x=0; x<ret->cols; x++)
                                        ret->bitmap[y][x] = 0;
                }
        }
        else {
                fld = im->deinterlace(1);
                for (y=0; y<ret->rows; y++) {
                        if (y%factor==0) {
                                for (x=0; x<ret->cols; x++)
                                        ret->bitmap[y][x] = fld->bitmap[z][x];
                                z += 1;
                        }
                        else
                                for (x=0; x<ret->cols; x++)
                                        ret->bitmap[y][x] = 0;
                }
        }
        delete im;
        //delete fld;
```

```
        return ret;

}


pixel getpixel(image &im, int x, int y)
{
        if (y<0 || y>im.rows-1)
                return 0;
        else
                return im.bitmap[y][x];
}



void showpixel(const image *im, char *msg, int top, int left, int bottom, int right)
{
        cout << "\n" << msg << "\n";
        for (int i=top; i<=bottom; i++) {
                for (int j=left; j<=right; j++)
                        cout << getpixel(*im, j, i) << "  ";
                cout << "\n";
        }
}



image *filtreVT(const image *past, const image *pres, const image *futur, int imp)
{
    filter dirac(1);
        int y, nbligne;
        pixel X=0;
        float D00=0,D10=0,D20=0, D01=0,D11=0,D21=0, D02=0,D12=0,D22=0;
        float D30=0,D31=0,D32=0,D40=0,D41=0,D42=0;

        if (imp) {
                y = 1;
                nbligne = pres->rows-1;
        }
        else {
                y = 0;
                nbligne = pres->rows-2;
        }

        image *im00 = new image(pres->rows, pres->cols);
        im00->copy(*pres);
        image *im0 = new image(pres->rows, pres->cols);
        im0->copy(*pres);
        image *im = new image(pres->rows, pres->cols);
        im->copy(*pres);


for (; y<=nbligne; y+=2)
        for (int x=0; x<=im->cols; x++) {
```

```
//      DETECTION DU CONTOUR

D00=abs(pres->pixread(x-3,y-1)+pres->pixread(x-2,y-1)+pres->pixread(x-1,y-1)
   +pres->pixread(x,y-1)+pres->pixread(x+1,y-1)+pres->pixread(x+2,y-1)
   +pres->pixread(x+3,y-1)-pres->pixread(x-3,y+1) -pres->pixread(x-2,y+1)
   -pres->pixread(x-1,y+1)-pres->pixread(x,y+1)-pres->pixread(x+1,y+1)
   -pres->pixread(x+2,y+1)-pres->pixread(x+3,y+1));
D01= abs(pres->pixread(x-4,y-1)+pres->pixread(x-3,y-1)+pres->pixread(x-2,y-1)
   +pres->pixread(x-1,y-1)+pres->pixread(x,y-1)+pres->pixread(x+1,y-1)
   +pres->pixread(x+2,y-1)-pres->pixread(x-4,y+1)-pres->pixread(x-3,y+1)
   -pres->pixread(x-2,y+1)-pres->pixread(x-1,y+1)-pres->pixread(x,y+1)
   -pres->pixread(x+1,y+1)-pres->pixread(x+2,y+1));
D02=abs(pres->pixread(x-2,y-1)+pres->pixread(x-1,y-1)+pres->pixread(x,y-1)
   +pres->pixread(x+1,y-1) +pres->pixread(x+2,y-1)+pres->pixread(x+3,y-1)
   +pres->pixread(x+4,y-1) -pres->pixread(x-2,y+1)-pres->pixread(x-1,y+1)
   -pres->pixread(x,y+1)-pres->pixread(x+1,y+1)-pres->pixread(x+2,y+1)
   -pres->pixread(x+3,y+1)-pres->pixread(x+4,y+1));


 D10=abs(pres->pixread(x-1,y-1)+pres->pixread(x,y-1)+pres->pixread(x+1,y-1)
      +pres->pixread(x+2,y-1)+pres->pixread(x+3,y-1)+pres->pixread(x+4,y-1)
      +pres->pixread(x+5,y-1)-pres->pixread(x-5,y+1)-pres->pixread(x-4,y+1)
      -pres->pixread(x-3,y+1) -pres->pixread(x-2,y+1) -pres->pixread(x-1,y+1)
       -pres->pixread(x,y+1) -pres->pixread(x+1,y+1));
 D11=abs(pres->pixread(x-2,y-1)+pres->pixread(x-1,y-1)+pres->pixread(x,y-1)
      +pres->pixread(x+1,y-1) +pres->pixread(x+2,y-1)+pres->pixread(x+3,y-1)
      +pres->pixread(x+4,y-1)-pres->pixread(x-6,y+1)-pres->pixread(x-5,y+1)
      -pres->pixread(x-4,y+1)-pres->pixread(x-3,y+1)-pres->pixread(x-2,y+1)
      -pres->pixread(x-1,y+1)-pres->pixread(x,y+1));
D12=abs(pres->pixread(x,y-1)+pres->pixread(x+1,y-1)+pres->pixread(x+2,y-1)
      +pres->pixread(x+3,y-1)+pres->pixread(x+4,y-1)+pres->pixread(x+5,y-1)
      +pres->pixread(x+6,y-1)-pres->pixread(x-4,y+1)-pres->pixread(x-3,y+1)
      -pres->pixread(x-2,y+1) -pres->pixread(x-1,y+1) -pres->pixread(x,y+1)
      -pres->pixread(x+1,y+1) -pres->pixread(x+2,y+1));
 D20=abs(pres->pixread(x-5,y-1)+pres->pixread(x-4,y-1)+pres->pixread(x-3,y-1)
      +pres->pixread(x-2,y-1)+pres->pixread(x-1,y-1)+pres->pixread(x,y-1)
      +pres->pixread(x+1,y-1)-pres->pixread(x+5,y+1)-pres->pixread(x+4,y+1)
      -pres->pixread(x+3,y+1)-pres->pixread(x+2,y+1)-pres->pixread(x+1,y+1)
      -pres->pixread(x,y+1)-pres->pixread(x-1,y+1));
 D21=abs(pres->pixread(x-6,y-1)+pres->pixread(x-5,y-1)+pres->pixread(x-4,y-1)
     +pres->pixread(x-3,y-1)+ pres->pixread(x-2,y-1)+pres->pixread(x-1,y-1)
      +pres->pixread(x,y-1) -pres->pixread(x+4,y+1)-pres->pixread(x+3,y+1)
      -pres->pixread(x+2,y+1)-pres->pixread(x+1,y+1) -pres->pixread(x,y+1)
     -pres->pixread(x-1,y+1)-pres->pixread(x-2,y+1));
 D22=abs(pres->pixread(x-4,y-1)+pres->pixread(x-3,y-1)+pres->pixread(x-2,y-1)
    +pres->pixread(x-1,y-1)+pres->pixread(x,y-1)+pres->pixread(x+1,y-1)
    +pres->pixread(x+2,y-1)-pres->pixread(x+6,y+1) -pres->pixread(x+5,y+1)
    -pres->pixread(x+4,y+1) -pres->pixread(x+3,y+1)-pres->pixread(x+2,y+1)
    -pres->pixread(x+1,y+1) -pres->pixread(x,y+1));


 D30=abs(pres->pixread(x-2,y-1)+pres->pixread(x-1,y-1)+pres->pixread(x,y-1)
      +pres->pixread(x+1,y-1)+pres->pixread(x+2,y-1)+pres->pixread(x+3,y-1)
```

```
      +pres->pixread(x+4,y-1)-pres->pixread(x-4,y+1)-pres->pixread(x-3,y+1)
      -pres->pixread(x-2,y+1) -pres->pixread(x-1,y+1) -pres->pixread(x,y+1)
      -pres->pixread(x+1,y+1) -pres->pixread(x+2,y+1));
 D31=abs(pres->pixread(x-3,y-1)+pres->pixread(x-2,y-1)+pres->pixread(x-1,y-1)
      +pres->pixread(x,y-1)+pres->pixread(x+1,y-1)+pres->pixread(x+2,y-1)
      +pres->pixread(x+3,y-1)-pres->pixread(x-5,y+1)-pres->pixread(x-4,y+1)
      -pres->pixread(x-3,y+1) -pres->pixread(x-2,y+1) -pres->pixread(x-1,y+1)
       -pres->pixread(x,y+1) -pres->pixread(x+1,y+1));
 D32=abs(pres->pixread(x-1,y-1)+pres->pixread(x,y-1)+pres->pixread(x+1,y-1)
      +pres->pixread(x+2,y-1)+pres->pixread(x+3,y-1)+pres->pixread(x+4,y-1)
      +pres->pixread(x+5,y-1)-pres->pixread(x-3,y+1)-pres->pixread(x-2,y+1)
      -pres->pixread(x-1,y+1) -pres->pixread(x,y+1) -pres->pixread(x+1,y+1)
       -pres->pixread(x+2,y+1) -pres->pixread(x+3,y+1));

 D40=abs(pres->pixread(x-2,y+1)+pres->pixread(x-1,y+1)+pres->pixread(x,y+1)
      +pres->pixread(x+1,y+1)+pres->pixread(x+2,y+1)+pres->pixread(x+3,y+1)
      +pres->pixread(x+4,y+1)-pres->pixread(x-4,y-1)-pres->pixread(x-3,y-1)
      -pres->pixread(x-2,y-1) -pres->pixread(x-1,y-1) -pres->pixread(x,y-1)
       -pres->pixread(x+1,y-1) -pres->pixread(x+2,y-1));
 D41=abs(pres->pixread(x-3,y+1)+pres->pixread(x-2,y+1)+pres->pixread(x-1,y+1)
      +pres->pixread(x,y+1)+pres->pixread(x+1,y+1)+pres->pixread(x+2,y+1)
      +pres->pixread(x+3,y+1)-pres->pixread(x-5,y-1)-pres->pixread(x-4,y-1)
      -pres->pixread(x-3,y-1) -pres->pixread(x-2,y-1) -pres->pixread(x-1,y-1)
       -pres->pixread(x,y-1) -pres->pixread(x+1,y-1));
 D42=abs(pres->pixread(x-1,y+1)+pres->pixread(x,y+1)+pres->pixread(x+1,y+1)
      +pres->pixread(x+2,y+1)+pres->pixread(x+3,y+1)+pres->pixread(x+4,y+1)
      +pres->pixread(x+5,y+1)-pres->pixread(x-3,y-1)-pres->pixread(x-2,y-1)
      -pres->pixread(x-1,y-1) -pres->pixread(x,y-1) -pres->pixread(x+1,y-1)
       -pres->pixread(x+2,y-1) -pres->pixread(x+3,y-1));



{
   if((D40<D00 && D40<D10&&D40<D30&&D40<D20)&&((D41<D01&&
D41<D11&&D41<D31&& D41<D21) &&(D42<D02 && D42<D12&&D42<D32&&D42<D22)))
    im->pixref(x,y)=6;
 else if((D30<D00 && D30<D20&&D30<D10&&D30<D40 )&&((D31<D01&&
D31<D21&&D31<D11&&
D31<D41)&&(D32<D02 && D32<D22&&D32<D12&&D32<D42 )))

    im->pixref(x,y)=1;

 else if((D20<D00 && D20<D10&&D20<D30&&D20<D40)&&((D21<D01&&
D21<D11&&D21<D31&&
D21<D41) &&(D22<D02 && D22<D12&&D22<D32&&D22<D42)))
    im->pixref(x,y)=186;
 else if((D10<D00 && D10<D20&&D10<D30&&D10<D40 )&&((D11<D01&&
D11<D21&&D11<D31&&
D11<D41)&&(D12<D02 && D12<D22&&D12<D32&&D12<D42 )))
    im->pixref(x,y)=31;
              else
    im->pixref(x,y)=0;
 }
}
```

```
//        FORTIFICATION DE LA DECISION

       pixel p,q,r,s;
      if (imp) y = 1;
      else y = 0;
      for (; y<=im->rows; y+=2)
      for (int x=0; x<=im->cols; x++)


 {
p=im->pixread(x,y)+im->pixread(x-4,y-2)+im->pixread(x+4,y+2); q=im->pixread(x,y)
+im->pixread(x+4,y-2)+im->pixread(x-4,y+2); r=im->pixread(x,y)+im->pixread(x-2,y
-2)+im->pixread(x+2,y+2); s=im->pixread(x,y)+im->pixread(x+2,y-2)+im->pixread(x-
2,y+2);

     if (372<=p)       im0->pixref(x,y)=186;
     if (62<=q&&q<=155) im0->pixref(x,y)=31;
     if (12<=r&&r<=30)  im0->pixref(x,y)=6;
     if (2<=s&&s<=5)    im0->pixref(x,y)=1;

    }
      if (imp) y = 1;
      else y = 0;
      for (; y<=im->rows; y+=2)
      for (int x=0; x<=im->cols; x++)
 {
 p=im0->pixread(x,y)+im0->pixread(x-1,y)+ im0->pixread(x+1,y);
     if (372<=p)       im00->pixref(x,y)=186;
     if (62<=p&&p<=155) im00->pixref(x,y)=31;
     if (12<=p&&p<=30)  im00->pixref(x,y)=6;
     if (2<=p&&p<=5)    im00->pixref(x,y)=1;


 }

//        INTERPOLATION SELON LE CONTOUR


      pixel p1;
      if (imp) y = 1;
      else y = 0;
      for (; y<=im->rows; y+=2)
      for (int x=0; x<=im->cols; x++)
 {
  p1=im00->pixread(x,y);

     if (p1==186)  X =
             (32*(past->pixread(x, y)   + futur->pixread(x, y)) +
              64*(pres->pixread(x-2, y-1) + pres->pixread(x+2, y+1)) -
              16*(past->pixread(x-4, y-2) + futur->pixread(x-4, y-2) +
                past->pixread(x+4, y+2) + futur->pixread(x+4, y+2)))/128;
```

```
    else  if (p1==31) X =
              (32*(past->pixread(x, y)   + futur->pixread(x, y)) +
              64*(pres->pixread(x+2, y-1) + pres->pixread(x-2, y+1)) -
              16*(past->pixread(x+4, y-2) + futur->pixread(x+4, y-2) +
                past->pixread(x-4, y+2) + futur->pixread(x-4, y+2)))/128;

    else  if (p1==6)   X =
             (32*(past->pixread(x, y)   + futur->pixread(x, y)) +
              64*(pres->pixread(x-1, y-1) + pres->pixread(x+1, y+1)) -
              16*(past->pixread(x-2, y-2) + futur->pixread(x-2, y-2) +
                past->pixread(x+2, y+2) + futur->pixread(x+2, y+2)))/128;

    else  if (p1==1)  X =
              (32*(past->pixread(x, y)   + futur->pixread(x, y)) +
              64*(pres->pixread(x+1, y-1) + pres->pixread(x-1, y+1)) -
              16*(past->pixread(x+2, y-2) + futur->pixread(x+2, y-2) +
                past->pixread(x-2, y+2) + futur->pixread(x-2, y+2)))/128;


    else   X=
             (32*(past->pixread(x, y)   + futur->pixread(x, y)) +
              56*(pres->pixread(x, y-1) + pres->pixread(x, y+1)) -
              16*(past->pixread(x, y-2) + futur->pixread(x, y-2) +
                past->pixread(x, y+2) + futur->pixread(x, y+2)) +
              8*(pres->pixread(x, y-3) + pres->pixread(x, y+3)))/128;


  im0->pixref(x,y) = X;

  }
delete im00;

//    CONVERSION EN FORMAT HDTV

      // vertical
      filter ver("cofHDTV21v.flt");
      image *tmp=ver.ivertical(*im0,2);  // sur-echantillonage et
                                  // filtrage.
      delete im0;
      image *out=dirac.vertical(*tmp,1);  // decimation par 1
      delete tmp;

      // horizontale
      filter hor("cofHDTV21.flt");
      tmp=hor.ihorizontal(*out,2);  // sur-echantillonage et
                                  // filtrage.
      delete out;
      out=dirac.horizontal(*tmp,1);  // decimation par 1
      delete tmp;
 // fin de horizontal

      for (y=0; y<= 480; y+=1)
      for (int x=0;x<= 720; x++)
```

```
  im->pixref(x,y) = out->pixread(x+360,y+240);

delete out;
      return im;
}


header *createheader(char *nm, definedType t, int seqlen, int intlc, int r, int c)
{

      header *h = new header(nm, t, seqlen, r, c);

      h->interlace = intlc;

      return h;
}


float SNR(const image *original, const image *im)
{
      float S=0, N=0;

      for (int i=0; i<im->rows; i++)
            for (int j=0; j<im->cols; j++)
                  S = S + pow((getpixel(*original, j, i)),2);


      image *tmp = new image(original->rows, original->cols);
      tmp->copy(*original);
      *tmp -= *im;
      for (i=0; i<im->rows; i++)
            for (int j=0; j<im->cols; j++)
                  N = N + pow((getpixel(*tmp, j, i)),2);

      float resultat = 10*log10(S/N);

      return resultat;
}
```

BIBLIOGRAPHIE

[1] Murat Kunt, Goesta Granlund, Michel Kocher, *Traitement numérique des images* , Presses Polytechniques, 1993.

[2] Yves C. Faroudja, "NTSC and BEYOND", *IEEE Transactions on Consumer Electronics*, vol 34, No 1, Feb 1988.

[3] Anil K. Jain, *Fundamentals of Digital Image Processing*, Prentice Hall, University of California, Davis, 569 pages, 1989.

[4] A. P. Pentland, " Local shading analysis", *IEEE Patt. Anal. Machine Intell.*, vol. 6 No. 2, pp 170-187, 1984.

[5] J. J. Koenderink, " Design for a sensorium", in W. von Seelen, B. Shaw, and U.M. Leinho (Eds), *Org. Neural Networks,* Verlagsgesellschaft mbH, pp 185-207, 1988.

[6] J. J. Koenderink and A. J. van Doorn, " Representation of local geometry in the visual system", *Biol. Cybern.,* vol. 55, pp 367-375.

[7] W. T. Freedman and Edward H. Adelson, " The Design and Use of Steerable Filters", *IEEE Patt. Anal. Machine Intell.*, vol 13, No. 9, pp 891-905, 1991.

[8] P. Danielsson and O. Serger, " Rotation invariance in gradient and higher order derivative detectors", *Comp. Vision Graphics Image Proccessing*, vol. 49, pp 198-221, 1990.

[9] W. T. Freeman and A. Jepson, " Computation of normal velocity from local phase information", in *Pro. IEEE CVPR (San Diego, CA),* pp 379-386, 1989.

[10] R. M. Haralick, " The digital step edge from zero crossings of second directional derivative", *IEEE Patt. Anal. Machine Intell.*, vol. 6, no 1, pp 58-68, 1984.

[11] M. Kass and A. Witkin, " Analyzing oriented patterns", *Comp. Vision Graphics Image Processing*, vol. 37, pp 362-385, 1987.

[12] A. Nguyen and E. Dubois, "Spatial directional interpolation using steerable filters," in *Proc. Canadian Conference on Electrical and Computer Engineering*, (Toronto, ON), pp. MA4.8.1-MA4.8.4, Sept. 1992.

[13] R. G. Keys, " Cubic Convolution Interpolation for Digital Image Processing", *IEEE Trans. Acoust. Speech Signal Process.*, vol., No. 6, pp 1153-1160, Dec. 1981.

[14] R. Simonetti, A. Polo Filisan, S. Carrato, " A Deinterlacer for IQTV Receivers and Multimedia Applications", IEEE Trans. Cons. Electronics, vol. 39, No. 3, pp 234-239, Aug 1993.

[15] D. M. Martinez and J. S. Lim, *Spatial Interpolation of Interlaced Television Pictures*, Research Laboratory of Electronic, MIT.

[16] M. Isnardi, *Modeling the Television Process*, Thèse de doctorat (Ph. D.), Massachusetts Institute of Technology, 1986.

[17] Gary A. Jones, Paul A. Snopko, Jong G. Kim, " DSC-HDTV Broading with Upconverted NTSC Video", *IEEE Transaction on Broadcasting,* vol. 38, No. 1, pp 7-11, Mars 1992.

[18] CCIR Recommendation 601-1, *Encoding Parameters of Digital Television for Studios*, International Telecommunication Union, Geneva, 1986.

[19] T. Reuter, " Standards Conversion Using Motion Compensation", *Signal Processing*, vol. 16, pp 73-82, Jan. 1989.

[20] Denis M Martinez, *Model Based Motion Estimation and its Application to Restoration and Interpolation of Motion Pictures* , Ph. D. thesis, Massachusetts Institute of Technology, 1986.

[21] Jae S. Lim, *Two dimensional signal and image processing*, Prentice Hall, pp 196, 1990.

[22] V. G. Devereux, *Standards Conversion Between 1250/50 and 625/50 TV Systems*, BBC Research Department UK, 51 pages, 1992.

[23] M. Weston, UK Patent Application GB 2 184628A, June 1987.

[24] Gary Tonge, " Image Processing for Higher Definition Television", *IEEE Transaction on Circuits and Systems*, vol. cas-34, pp1385-1397, Nov. 1987.

[25] John Watkinson, *The Art of Digital Video*, Focal Press, 1990.

[26] C. T. Ledinh, *Traitements avancés des signaux*, Notes de cours GEI-752, Département de génie électrique de l'Université de Sherbrooke, automne 1994.

[27] Yrjö, S. D. C. Yu, and S.K. Mitra, " Interpolated Finite Impulse Response Filters", *IEEE Transaction on Acoutics, Speech, and Signal Processing*, vol. ASSP-32, no 3, June 1984.